LUCAS YUJI HARADA

PoliLab - Sistema de gerenciamento de dispositivos IoT para laboratório educacionais

São Paulo
2021

LUCAS YUJI HARADA

PoliLab - Sistema de gerenciamento de dispositivos IoT para laboratório educacionais

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção de Título de Engenheiro Eletricista com ênfase em Computação.

São Paulo
2021

LUCAS YUJI HARADA

PoliLab - Sistema de gerenciamento de dispositivos IoT para laboratório educacionais

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção de Título de Engenheiro Eletricista com ênfase em Computação.

Orientador:
Reginaldo Arakaki

Co-orientadores:
Victor Takashi Hayashi
Felipe Valencia de Almeida

São Paulo
2021

AGRADECIMENTOS

Agradeço a meus orientadores pela grande ajuda durante o desenvolvimento deste trabalho, ao meu amigo Marco Abensur por sua *expertise* em eletrônica e me ajudar com a realização da entrevista com a equipe da Fieldpro, e a meus pais pelo apoio emocional e financeiro durante a minha graduação.

RESUMO

Estudantes de engenharia aprendem uma variedade de conhecimentos teóricos em salas de aula, mas para se obter conhecimento prático, o uso de laboratórios educacionais é imprescindível. A restrição de circulação derivada da pandemia do COVID-19 impossibilitou o acesso a tais laboratórios, resultando na necessidade de se viabilizar o controle e gerenciamento remoto dos equipamentos dos laboratórios. Dentro desse contexto, a modernização de equipamentos convencionais de laboratórios através de dispositivos de Internet das Coisas (IoT) inteligentes pode ser uma solução viável, e traz consigo vários benefícios decorrentes da viabilização do acesso remoto aos laboratórios educacionais como acessibilidade e redução de custos. O projeto PoliLab é um sistema de gerenciamento de dispositivos IoT para que funcionários de instituições educacionais possam comandar e monitorar remotamente o uso e consumo energético de equipamentos de laboratórios através de uma interface Web com métricas coletadas em tempo real.

Palavras-Chave – Internet das Coisas, Laboratórios educacionais

ABSTRACT

Engineering students learn a variety of theoretical knowledge in classrooms. class, but to obtain practical knowledge, the use of educational laboratories is essential. The lockdowns originated from the COVID-19 pandemic made accessing such laboratories impossible, resulting in the need to enable remote control and management of laboraty equipments. Within this context, the modernization of conventional laboratory equipment through intelligent Internet of Things (IoT) devices can be a viable solution, and brings with it several benefits arising from the use of remote access to educational laboratories such as accessibility and cost reductions. The PoliLab project is an IoT device management system for employees of educational institutions to remotely command and monitor the use and energy consumption of laboratory equipment through a web interface with metrics collected in real time.

Keywords – Internet of Things, Educational laboratories

LISTA DE FIGURAS

1 Evolução do número de publicações relacionadas à IoT ao longo do tempo. 12

2 Uso relativo de energia entre uma transmissão usando TLS, e outra não usando nenhuma criptografia. 17

3 Número de publicações relacionadas a laboratórios remoto. 20

4 Ficha de Osterwalder 24

5 As 5 visões do modelo RM-ODP e suas interdependências. 26

6 Cliente (esquerda) realizando uma requisição HTTP a um servidor web (direita). 27

7 *Publishers* (esquerda) publicando mensagens no tópico <id>/power a um *broker* (meio), com *subscribers* (direita) consumindo apenas as mensagens que os interessam. 28

8 Cliente acessando recurso Pet com ID 1 com resposta serializada em JSON de uma API REST. 31

9 Cliente acessando campos específicos do recurso Pet com ID 1, juntamente com o campo relacionado "owner"via GraphQL 32

10 Arquitetura do LabEAD. 34

11 Ficha de Osterwalder do projeto implementado neste trabalho. 35

12 Hierarquia dos cargos da aplicação 40

13 Hierarquia de um laboratório fictício 42

14 Arquitetura obtida com após divisão em subsistemas 44

15 Cliente realizando fluxo de login com o subsistema de autenticação 44

16 Fluxo de uma requisição dentro do Hasura usando 3 tipos de webhooks. . . 45

17 Foto do hardware do dispositivo IoT instalado no laboratório. 49

18 Arquitetura e tecnologias usadas nos subsistema do PoliLab 51

19 Comparativo de taxa de ingestão de dados de séries temporais entre PostgreSQL e TimescaleDB 52

20 Tela de autenticação da aplicação PoliLab 53

21 Calendário de eventos programados para o laboratório "Lab1"da aplicação PoliLab. 54

22 Consumo energético dos últimos 7 dias do laboratório "Lab1"da aplicação PoliLab. 54

23 Métricas de consumo energético do dispositivo "c88e35"em tempo real, monitorado pela aplicação PoliLab 55

24 Tela de criação de laboratório da aplicação PoliLab 56

25 Formulário para criação de um novo laboratório da aplicação PoliLab . . . 57

26 Formulário para registro de um novo dispositivo na aplicação PoliLab . . . 58

27 Calendário de eventos programados da aplicação PoliLab. No momento se encontra vazio pois não há comandos registrados. 58

28 Agendando um comando para desligar todos aparelhos do laboratório todos os dias às 18 horas na aplicação PoliLab. 59

29 Calendário da aplicação PoliLab atualizado com o novo comando registrado. 60

30 Tentativa de criação de comando bloqueada devido a falta de privilégios necessários por parte do usuário 60

LISTA DE TABELAS

1 Consumo de corrente por componente e estado no dispositivo Fieldpro . . . 18

2 As 4 possíveis categorias de laboratórios de acordo com (BENCOMO, 2004) 22

3 Mapeamento de requisitos funcionais e não-funcionais com o sistema que os implementa 39

SUMÁRIO

1 Introdução **11**

1.1 Motivação 11

1.2 Objetivo 12

1.3 Organização do trabalho 12

2 Aspectos conceituais **13**

2.1 IoT 13

2.1.1 Definição 13

2.1.2 Estado da arte 14

2.1.3 Desafios 15

2.1.4 Consumo energético 16

2.1.4.1 Aplicação prática na indústria - Fieldpro 18

2.2 Laboratório EAD 19

2.3 Ficha de Osterwalder 23

2.4 Arquitetura - RF, RNF, ODP (Visões) 24

2.4.1 RM-ODP (Visões) 25

2.5 Protocolos de comunicação 26

2.5.1 HTTP 27

2.5.2 MQTT 27

2.6 Bancos de dados 28

2.6.1 Relacionais (SQL) 28

2.6.2 Não-Relacionais (NoSQL) 29

2.6.3 Bancos de dados temporais 30

2.7 Web API 30

2.7.1 REST 30
2.7.2 GraphQL 31

3 PoliLab - Sistema de gerenciamento de dispositivos IoT para laboratórios educacionais 33

3.1 Contextualização e definição do problema 33
3.2 Ficha de Osterwalder 35
3.3 Visões Arquiteturais - RM-ODP 36
3.3.1 Visão de negócio 36
3.3.2 Visão de requisitos funcionais 37
3.3.3 Visão de requisitos não-funcionais 38
3.3.4 Visão de mecanismos de engenharia 38
3.3.4.1 Subsistema de autenticação 38
3.3.4.2 Subsistema de autorização 39
3.3.4.3 Subsistema de comandos 41
3.3.4.4 Subsistema de agendamento 42
3.3.4.5 Subsistema de monitoramento 43
3.3.4.6 Arquitetura proposta 43
3.3.5 Visão de tecnologia 43
3.3.5.1 Subsistema de autenticação 43
3.3.5.2 Subsistema de autorização 45
3.3.5.3 Subsistema de comandos 46
3.3.5.4 Subsistema de agendamento 47
3.3.5.5 Subsistema de monitoramento 48
3.4 Processo de implementação 49

4 Resultados 51

4.1 Configuração da aplicação 55

5 Conclusão **61**

5.1 Sobre o objetivo do projeto de TCC 61

5.2 Em relação ao curso de engenharia 62

5.3 Possíveis evoluções do produto 63

Referências **64**

Apêndice A – Arquivos de configuração **68**

1 INTRODUÇÃO

1.1 Motivação

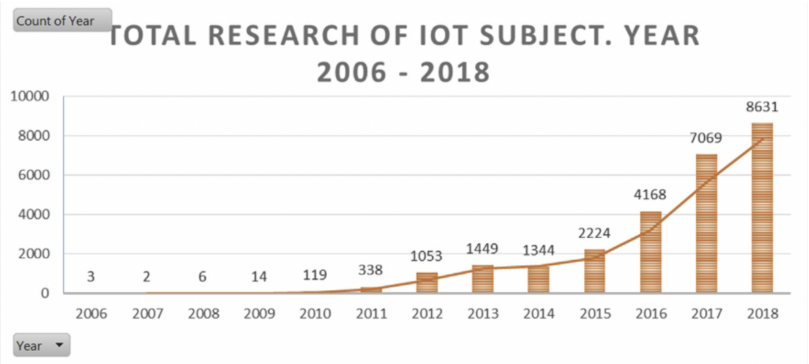
Desde a antiguidade, a humanidade tem realizado experimentos para explorar problemas, sejam eles de natureza cotidiana ou científica. O mestre filósofo Aristóteles acreditava na necessidade de se juntar o conhecimento teórico (intelecto) com a prática (ação) para fixar o aprendizado de alunos (CRUZ; CAMPOS, 2016), e esse sentimento ainda persiste nos educadores dos tempos modernos. Nesse contexto, os laboratórios educacionais são imprescindíveis na habilitação do aprendizado prático no currículo educacional dos discentes.

No Brasil, de acordo com o Censo Escolar de 2018 (INEP CENSO, 2018), apenas 37,5% das escolas da rede estadual tem acesso a laboratórios educacionais, e laboratórios remotos representam uma oportunidade de mudar esse panorama. Laboratórios remotos tem vários benefícios em relação aos laboratórios locais (HAYASHI et al., 2020) como (a) a redução de custos (tanto por parte das instituições educacionais quanto dos alunos), pois permitem comportar um maior número de alunos e compartilhar recursos entre escolas e universidades diferentes, (b) maior disponibilidade, pois ambientes onlines permitem maior flexibilidade de horários, (c) maior acessibilidade, pois permitem acessar os recursos e equipamentos de um laboratório sem necessitar a locomoção dos alunos, principalmente daqueles com alguma deficiência locomotora, e (d) observabilidade, pois permitem monitorar de perto o progresso dos alunos individualmente.

Devido a pandemia do COVID-19 e as restrições circulatórias originadas das medidas de contenção do vírus, o acesso local de praticamente todas instituições educacionais foi interrompido, e isso afetou negativamente principalmente as disciplinas que requerem utilizar um laboratório educacional de acesso local. Concomitantemente a essa crise, o interesse acadêmico na última década em soluções de Internet das Coisas (IoT, do inglês "Internet of Things") tem crescido (Figura 1). A Internet das Coisas possibilita um mundo de novas tecnologias nunca antes vistas, mas mais importante do que isso, permite trazer

inteligência e conectividade a aparelhos e equipamentos convencionais.

Figura 1: Evolução do número de publicações relacionadas à IoT ao longo do tempo.



Fonte: (DACHYAR; ZAGLOEL; SARAGIH, 2019)

A motivação para dar início a este projeto foi uma junção de todos os fatores mencionados anteriormente.

1.2 Objetivo

O objetivo deste trabalho é criar um sistema de software para facilitar a integração de dispositivos IoT com laboratórios educacionais de forma que isso possibilite transformar laboratórios convencionais em laboratórios de acesso remoto. Isso será feito trazendo inteligência para os equipamentos convencionais dos laboratórios através de IoT para que seja possível melhorar a experiência de aprendizado remoto. Por se tratar de um projeto IoT, será necessário utilizar tanto componentes de hardware quanto de software, porém o escopo do projeto irá incluir apenas o lado do software, mais especificamente, a implementação de uma dashboard de gerenciamento de dispositivos IoT.

1.3 Organização do trabalho

O capítulo 2 introduz aspectos essenciais ao entendimento do projeto e o estado da arte no campo de IoT e laboratório remotos. O capítulo 3 mostra a aplicação de técnicas de design da arquitetura do projeto, partindo de primeiros princípios do problema inicial e culminando no design final. O capítulo 4 apresenta a solução implementada, e também demonstra rapidamente como seria a configuração da aplicação para um novo usuário. O capítulo 5 traz as conclusões e lições aprendidas durante o projeto e as possíveis abordagens para dar continuidade ao mesmo.

2 ASPECTOS CONCEITUAIS

2.1 IoT

2.1.1 Definição

O termo "Internet of Things"(IoT), ou traduzido do inglês "Internet das Coisas", foi usado pela primeira vez em 1999 por Kevin Ashton durante sua pesquisa na área de *Radio Frequency Identification* (RFID) sob a Procter&Gamble (P&G)(ASHTON et al., 2009), e desde então foi rapidamente adotado pela indústria e pela academia. A definição mais comumente usada para IoT é simplesmente "uma rede de objetos físicos". (HALLER; KARNOUSKOS; SCHROTH, 2008) define IoT como "um mundo onde objetos físico estão perfeitamente integrados com a rede de informações, e onde objetos físicos são participantes ativos em processos de negócio". (PATEL; PATEL et al., 2016) define IoT como "... uma internet de três componentes: **1.** de pessoas para pessoas, **2.** de pessoas para máquinas ou coisas, **3.** de máquinas ou coisas para outras máquinas ou coisas". A definição de cada autor foca em aspectos diferentes do IoT: enquanto (HALLER; KARNOUSKOS; SCHROTH, 2008) está mais focado no impacto que IoT tem sobre negócios, (PATEL; PATEL et al., 2016) foca na interação humano-computador; porém ambos convergem na presença computadores interagindo com processos humanos.

Apesar do termo apenas ter sido definido em 1999, antes dessa data já existiam visionários integrando produtos convencionais para controle remoto via Internet como levantado por (SURESH et al., 2014) em sua análise histórica de IoT. John Romkey foi um desses visionários, conectando sua torradeira a Internet em 1990. Em 1994 Steve Mann, pai da Computação Vestível, criou a WearCam, uma camêra vestível e conectada a internet. Embora hajam famílias de dispositivos IoT que nunca atingiram penetração no mercado, hoje, depois de mais de 20 anos de progresso, pode-se afirmar que humanos e a Internet das Coisas são quase inseparáveis: basta verificar a enorme parcela da população que utiliza aparelhos celulares diariamente.

2.1.2 Estado da arte

Como IoT se trata de um amplo campo de estudo, o estado da arte abrange colaborações com os mais diversos campos de pesquisa.

No campo de Big Data, dispositivos IoT são uma das principais fontes de dados responsáveis por gerar o volume de métricas necessários para se utilizar métodos estatísticos como Machine Learning. Devido a natureza de tais dispositivos, é comum a emissão de dados em tempo real que devem ser processados com baixa latência, e a partir desses requisitos, se identificou a necessidade da criação de métodos como Edge Computing.

No campo de Machine Learning existem várias aplicações na área de segurança onde dispositivos IoT são alvos de ataque ou são os atacantes em si; por exemplo (VERMA; RANGA, 2020) onde se realiza análises estatísticas para detecção de intrusos invadindo dispositivos IoT, e também (SHAFIQ et al., 2020) onde se analisa anomalias no tráfico de pacotes para detectar dispositivos IoT maliciosos dentro de uma rede. Os dados coletados por dispositivos IoT também podem ser utilizados para realizar autenticação de usuários baseado em seu comportamento rotineiro dentro de uma casa inteligente, como demonstrado por (HAYASHI; RUGGIERO, 2020).

No campo de Edge Computing, definido por (SHI et al., 2016) como "tecnologias que possibilitam computações a serem realizadas nas bordas da rede, na jusante de serviços de nuvem, e na montante de serviços IoT", existe uma clara sinergia entre as disciplinas pois o uso de técnicas de Edge Computing possibilita reduzir a latência no processamento do grande volume de dados gerados por dispositivos IoT, além de permitir reduzir a quantidade de banda e espaço de armazenamento consumidos nos serviços em nuvem.

No campo de Blockchain, de acordo com (REYNA et al., 2018), a possibilidade de se ter confiabilidade nos dados gerados por dispositivos IoT distribuídos sem a necessidade de uma autoridade central, além da rastreabilidade dos dados até sua fonte, pode impulsionar esforços como o da *Smart city*. Cidades Inteligentes são baseadas no conceito de dispositivos autônomos se comunicando entre si, e o Blockchain pode melhorar a autonomia dos dispositivos IoT já que facilita esforços de coordenação e confiabilidade de dados via o conceito do *ledger* distribuído. Sabe-se que uma das maiores preocupações que o público tem sobre o uso do Blockchain é o consumo energético que a mineração de criptomoedas das plataformas que dependem de *proof of work* (prova de trabalho). O uso de sistemas IoT como exemplificado no trabalho de (HAYASHI; ALMEIDA; KOMO, 2021) para monitorar tal consumo abre portas para futuramente implementar sistemas que o regularão.

2.1.3 Desafios

Como o campo de IoT é muito diverso, existem múltiplas perspectivas sobre os desafios que a área enfrenta atualmente. Os principais pontos identificados pelos trabalhos de (CHEN et al., 2014) e (FARHAN et al., 2017) foram:

a) Falta de padronização

Sistemas IoT são muito heterogêneos por existir a necessidade de atender casos de usos completamente diferentes, além de terem uma predisposição a serem sistemas distribuídos e descentralizados, e isso traz grandes complicações para realizar a integração de dados e a comunicação entre sistemas, pois cada solução distintas certamente adota protocolos e formatações de dados diferentes para atender seus requisitos.

b) Consumo energético

O consumo energético de um dispositivo é um fator determinante do seu tempo de vida para dispositivos que não tem acesso a uma fonte constante de energia. É comum terem aplicações IoT onde uma razão entre o tempo ativo e o tempo dormente de 1:1.000.000, e portanto é ideal que o vazamento de energia em modo dormente seja minimizado, mas isso é um problema difícil quando se tem circuitos CMOS (*Complementary metal–oxide–semiconductor*) com *designs* complexos, e muitas vezes é um requisito conflitante com o desafio abaixo.

c) Custo

Como a visão ideal para muitas aplicações IoT é ter uma distribuição em massa de seus dispositivos, é importante que o custo de fabricação de cada circuito individual seja baixo. Existe um *tradeoff* entre custo e desempenho, e pior desempenho geralmente implica em maiores latência de processamento, o que por sua vez implica em um aumento no tempo ativo do circuito, e portanto, resulta em um aumento do consumo energético, conflitando com o desafio mencionando anteriormente.

d) Tolerância a falhas

A comunicação entre dispositivos IoT pode ser interrompida pelas mais diversas maneiras como interferências ambientais, falta de energia, danos físicos, bloqueios na rede, entre outros. Como aplicações IoT costumam ser uma rede distribuída com múltiplos pontos de falha, é inaceitável que a falha de um dispositivo específico cause uma falha na tarefa que a rede como um todo está tentando realizar, e desse

modo, é importante a aplicação de técnicas de engenharia como demonstrado em (ARAKAKI; HAYASHI; RUGGIERO, 2020) para aumentar a disponibilidade de tais sistemas.

e) Privacidade

Existem muitos dispositivos IoT em uso hoje, e por conta disso empresas são capazes de coletar dados suficientes a ponto de se criar um perfil individual de cada usuário, que geralmente é utilizado para realizar decisões de negócio. Entretanto, nas mãos erradas, a existência desses dados é uma ameaça a privacidade do indivíduo, e por esse e outros motivos entidades governamentais criaram legislações como a Lei Geral de Proteção de dados (BRASIL, 2018) e a General Data Protection Regulation (EUROPEAN COMMISSION, 2018).

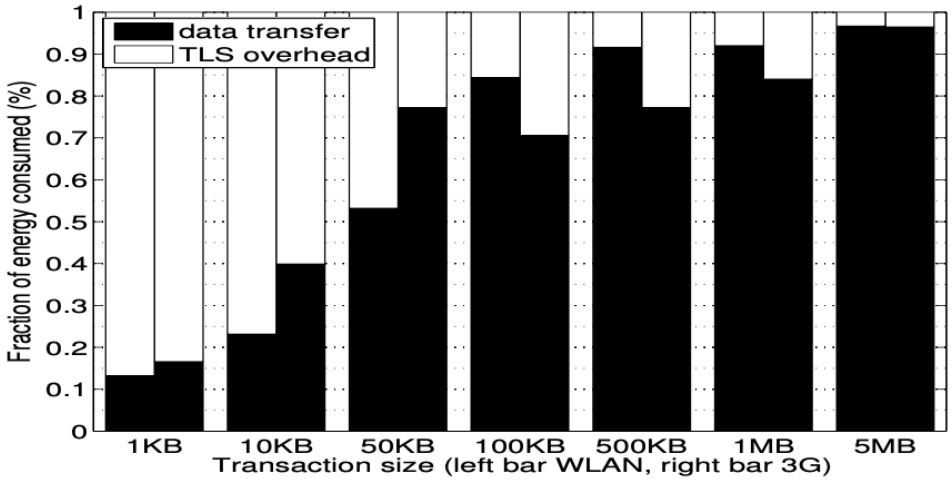
f) Segurança

Os ambientes operacionais que dispositivos IoT estão sujeitos são os mais diversos, mas comumente esses dispositivos se encontram expostos à internet, em redes públicas, ou acessíveis fisicamente por pessoas não autorizadas; uma visita rápida para websites (como https://shodan.io) que agregam dispositivos expostos publicamente de forma insegura é suficiente para confirmar a extensão do descuido geral. Além de serem alvos de ataque, dispositivos IoT também podem ser atacantes: um dispositivo infectado entrando dentro de uma rede interna pode ser um ponto de entrada para realização de escalonamento de privilégios. Outro ponto é o custo energético extra que algoritmos e protocolos criptográficos demandam dos dispositivos. A Figura 2 mostra que o *overhead* do uso do TLS (*Transport Layer Security*) para transmissões de dados pequenos ($<$ 1KB) é superior a 80% do total de energia necessário para fazer a mesma transmissão sem nenhuma criptografia. Considerando que muitas aplicações utilizam *payloads* pequenos para diminuir o tempo ativo do dispositivo e reduzir a banda consumida, geralmente a quantidade de dados transmitidos é bem inferior a 10KB.

2.1.4 Consumo energético

Dispositivos IoT tem tido um papel interessante no aumento da eficiência energética na indústria: o Google utiliza Machine Learning e IoT para controlar de maneira preditiva o uso do ar condicionado em seus *datacenters*, reduzindo o consumo energético em 40%; fábricas inteligentes utilizam IoT para regular dinâmicamente a carga de motores elétricos

Figura 2: Uso relativo de energia entre uma transmissão usando TLS, e outra não usando nenhuma criptografia.



Fonte: (MIRANDA; SIEKKINEN; WARIS, 2011)

de acordo com a flutuação do preço da energia; *smart grids* utilizam sensores e medidores inteligentes para monitorar e fornecer energia da maneira mais eficiente.

Dito isso, também é importante que os circuitos encontrados em dispositivos IoT sejam eficientes, e existem alguns fatores importantes que regem o consumo energético de um circuito. Um desses fatores é o projeto do circuito em si, que é influenciado principalmente pelas funcionalidades que o dispositivo irá realizar: um dispositivo que requer cálculos complexos para obter uma métrica obviamente irá consumir mais energia do que outro que apenas faz simples medição de um sensor. Outro fator é performance e qualidade dos componentes utilizados, que são inversamente relacionados com o custo de produção. Quanto menor for a performance de um microcontrolador, mais tempo o dispositivo terá que ficar no estado ativo para realizar as operações necessárias, e isso incorre em um maior consumo de energia. Um componente de má qualidade também pode ter maior vazamento de corrente, consumindo energia desnecessariamente. O uso de protocolos e algoritmos criptográficos, como mencionado na seção de Desafios, também são um fator do consumo de energia de um dispositivo. É indiscutível que essas técnicas são necessárias para realizar uma comunicação segura, mas infelizmente trazem um *overhead* significativo de energia como visto na Figura 2. Um último fator é a razão entre o tempo ativo e o tempo dormente do dispositivo. O consumo de energia durante os instantes de tempo que o circuito está ativo costuma ser da ordem de 10x a 1000x maior do que durante os estados dormentes, e minimizar o tempo que o circuito opera para apenas quando for estritamente necessário pode representar um grande ganho para a redução de consumo energético do dispositivo. Existem técnicas específicas que podem ser úteis como

o uso de *Pulse Width Modulation* (PWM) em circuitos com carga inercial (como motores elétricos) e o chaveamento de seções do circuito que não são de uso obrigatório durante certas operações, mas isso dependerá do circuito sob analise e também do ambiente de operação dele.

2.1.4.1 Aplicação prática na indústria - Fieldpro

Para o desenvolvimento mais concreto desta seção do trabalho, foi realizada uma entrevista com o time de engenharia da Fieldpro. A Fieldpro (FIELDPRO, 2021) é uma empresa que produz sensores meteorológicos IoT para monitoramento de métricas relevantes em lavouras, como a temperatura local, umidade do solo, velocidade do vento, milímetros de chuva, entre outros. Como o produto deles opera em localizações hostis (geralmente no meio da lavoura, expostos aos elementos) onde não é possível receber energia de fontes externas, e cuja conectividade varia muito dependendo da região geográfica, é preciso minimizar o consumo energético a todo custo para atingir o requisito operacional de 1 ano antes de ser necessário fazer uma recarga ou troca de bateria.

A Tabela 1 traz de forma simplificada o consumo energético de alguns dos componentes encontrados no dispostivo Fieldpro, e serve de contraste entre a quantidade de corrente que um dispositivo IoT consome durante o modo dormente e o modo ativo (coletando ou enviando métricas).

Tabela 1: Consumo de corrente por componente e estado no dispositivo Fieldpro

	microcontrolador	LED	comunicador
sleep	1	0	0
lendo sensores	20	0	0
enviando dados via Sigfox	20	15	20
enviando dados via Wi-Fi	20	15	100
enviando dados via 2g/3g	20	15	150

Fonte: Autoria própria

Números como esses servem para ressaltar a importância de se minimizar o tempo ativo do circuito, e reforça a necessidade de se utilizar utilizar técnicas como o chaveamento de partes do circuito (note como o comunicador não consome corrente em situações que não é preciso transmitir dados) para minimizar o consumo energético. Como a granularidade das métricas desejadas é apenas de hora em hora, o dispositivo Fieldpro pode ficar em modo *sleep* 99% do tempo, e apenas se ativar quando for necessário realizar uma medição, ou enviar os dados para casa. Para fim ilustrativo, caso não fosse feito a regulação do

tempo ativo do circuito, o tempo de vida do dispositivo diminuiria de um ano para apenas três dias.

2.2 Laboratório EAD

O estudo de engenharia é definido pela *Accreditation Board for Engineering and Technology* (ABET) como "Uma profissão na qual o conhecimento das ciências matemáticas e naturais adquirido pelo estudo, experiência e prática é aplicado com julgamento para desenvolver maneiras de se utilizar, economicamente, materiais e forças da natureza para o benefícios da humanidade". Estudantes de engenharia aprendem uma variedade de conhecimentos teóricos em salas de aula, mas para se obter conhecimento prático, o uso de laboratórios experimentais é imprescindível. De acordo com (FEISEL; ROSA, 2005), existem 3 principais categorias de laboratórios:

a) Laboratórios de desenvolvimento

   São laboratórios em que engenheiros visitam para obter dados experimentais a serem usados em algum projeto, ou validar se um projeto desempenha da maneira projetada. Há um foco em responder questões correntes.

b) Laboratórios de pesquisa

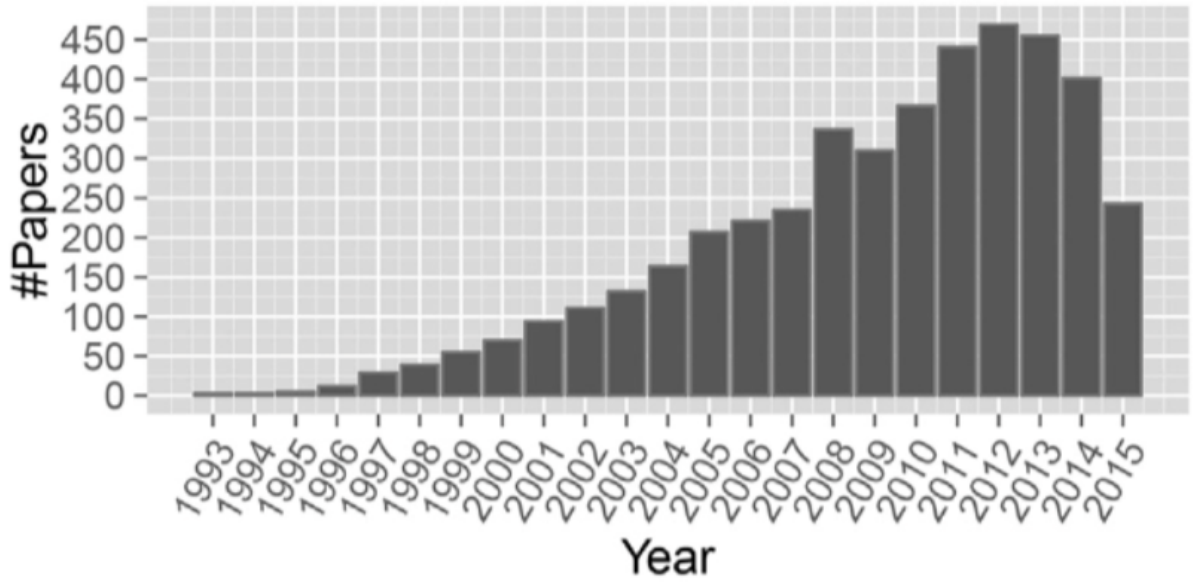
   São laboratórios que buscam expandir os horizontes do conhecimento humano de maneira sistematizada, sem necessariamente haver um uso imediato para as descobertas encontradas.

c) Laboratórios educacionais

   São laboratórios onde estudantes tem a oportunidade de aprender e colocar em prática os conhecimentos que aprenderam durante o curso. Tem um foco exploratório pela perspectiva dos alunos.

Como a demanda que originou essa dissertação surgiu de um laboratório educacional, este trabalho focará apenas nos laboratórios deste categoria. Desde 1996 já existia interesse acadêmico em laboratórios remotos, como demonstrado pelo trabalho de (AKTAN et al., 1996), onde se utilizou uma *webcam* e uma interface gráfica para interagir remotamente com um braço robótico via Internet no Laboratório de Engenharia de Controle da Universidade de do estado de Oregon.

Figura 3: Número de publicações relacionadas a laboratórios remoto.



Fonte: (HERADIO et al., 2016)

Sabendo da importância que os laboratórios têm na educação de um engenheiro e vendo um interesse crescente em educação remota (vide Figura 3), em 2002 a ABET já se preocupava com a qualidade do aprendizado remoto de engenharia. Havia (e ainda há) uma discussão sobre se laboratórios remotos eram capazes de cumprir os mesmos objetivos que laboratórios educacionais convencionais fornecem, mas até então não havia nenhuma diretriz sobre qual realmente é o objetivo de um laboratório educacional.

Por conta disso a ABET organizou um colóquio acadêmico, e depois de dois dias e meio de discussões, o grupo definiu uma lista dos treze principais aprendizados que alunos devem encontrar em um laboratório educacional (FEISEL; PETERSON, 2002):

a) Instrumentação

   Utilizar sensores, instrumentos, ou programa apropriados para coletar medidas físicas quantitativas.

b) Modelos

   Identificar os pontos fortes e limitações de modelos teóricos como preditores de comportamentos no mundo real.

c) Experimentos

Fazer o projeto de um experimento, especificar equipamentos e procedimentos apropriados, realizar tais procedimentos e interpretar os resultados obtidos.

d) Análise de dados

Demonstrar a habilidade de coletar, analisar e interpretar dados para formar e apoiar conclusões. Ser capaz de julgar ordens de magnitude e conhecer sistemas de unidade.

e) Projeto

Projetar, construir ou montar peças, produtos ou sistemas usando metodologias, equipamentos ou materiais específicos. Atender requisitos de clientes e desenvolver especificações de sistemas a partir desses requisitos. Testar e depurar protótipos, sistemas ou processos usando as ferramentas apropriadas para satisfazer os requisitos.

f) Aprender com as falhas

Reconhecer resultados incorretos devido a equipamentos, partes, códigos, processos ou projetos falhos para então reconstruir soluções efetivas.

g) Criatividade

Demonstrar níveis apropriados de independência intelectual, criatividade e capacidade de resolução de problemas do mundo real.

h) Psicomotor

Demonstrar competência na seleção, modificação e operação de ferramentas de engenharia.

i) Segurança

Reconhecer questões de saúde, segurança e meio ambiente relacionadas aos processos tecnológicos e atividades, e lidar com elas com responsabilidade.

j) Comunicação

Comunicar de modo efetivo sobre o trabalho realizado no laboratório para qualquer tipo de audiência, indo de resumos executivos até relatórios técnicos.

k) Trabalho em equipe

Trabalhar com eficácia em equipe, tendo responsabilidades individuais e conjuntas. Designar papéis, responsabilidades e tarefas, monitorar progresso, atingir prazos e integrar contribuições individuais em um único entregável.

Tabela 2: As 4 possíveis categorias de laboratórios de acordo com (BENCOMO, 2004)

		Tipo de acesso	
		Local	Remoto
Tipo de recurso	Real	acesso local, recurso real	acesso remoto, recurso real
	Simulado	acesso local, recurso simulado	acesso remoto, recurso simulado

Fonte: (BENCOMO, 2004)

l) Ética

Se comportar de maneira ética, reportar informações de maneira objetiva e agir com integridade.

m) Consciência sensorial

Usar os sentidos humanos para coletar informações e fazer julgamentos sólidos de engenharia e formular conclusões sobre questões do mundo real.

Como argumentado por (BALAMURALITHARA; WOODS, 2009), para boa parte desses objetivos, laboratórios remotos são capazes cumpri-los tão bem quanto um laboratório físico, mas é difícil que um ambiente remoto seja capaz de dar aos alunos a mesma qualidade de aprendizado psicomotor e de consciência sensorial que laboratórios locais. Além de cumprir a maiorias dos objetivos estabelecidos pela ABET, laboratórios remotos trazem certos benefícios extras (GRAVIER et al., 2008). Um deles é o aumento da disponibilidade, pois possibilitam o uso do laboratório a qualquer horário, por pessoas em regiões geográficas distantes, e até mesmo de universidades diferentes. Outro benefício é a observabilidade, pois todas interações podem ser registradas e gravadas já que são intermediadas por um sistema de software. Para experimentos que possuem algum nível de perigo, laboratórios remotos oferecem uma ambiente mais seguro para os alunos.

Laboratórios remotos podem ser classificados usando dois critérios, o método de acesso (local ou remoto) e o tipo de recurso (real ou simulado), conforme apresentado pela Tabela 2.

Laboratórios de acesso local com recursos reais são os laboratórios convencionais, onde o aluno está fisicamente presente para interagir com o recurso. Laboratórios de acesso local com recursos simulados tem seus experimentos executados em um ambiente completamente simulado por software encontrado dentro de uma das máquinas do laboratório, certos laboratórios da USP como o LARC (Laboratório de Arquitetura e Redes de Computadores) empregam essa modalidade. Laboratórios de acesso remoto com recursos reais são laboratórios em que o usuário opera equipamentos físicos através de uma interface

virtual, geralmente com o *feedback* sendo feito via vídeo através da Internet. Trabalhos como de (HAYASHI; HAYASHI, 2020) se enquadram nessa categoria. Por fim, laboratórios de acesso remoto com recursos simulados são laboratórios onde todo o ambiente é virtual, podendo ser acessados diretamente das casas dos alunos. Como é uma categoria ampla, pode ser implementado utilizando sistemas de realidade virtual (GEORGIOU; DIMITROPOULOS; MANITSARIS, 2007) ou motores de simulação via web (ACHUTHAN et al., 2011).

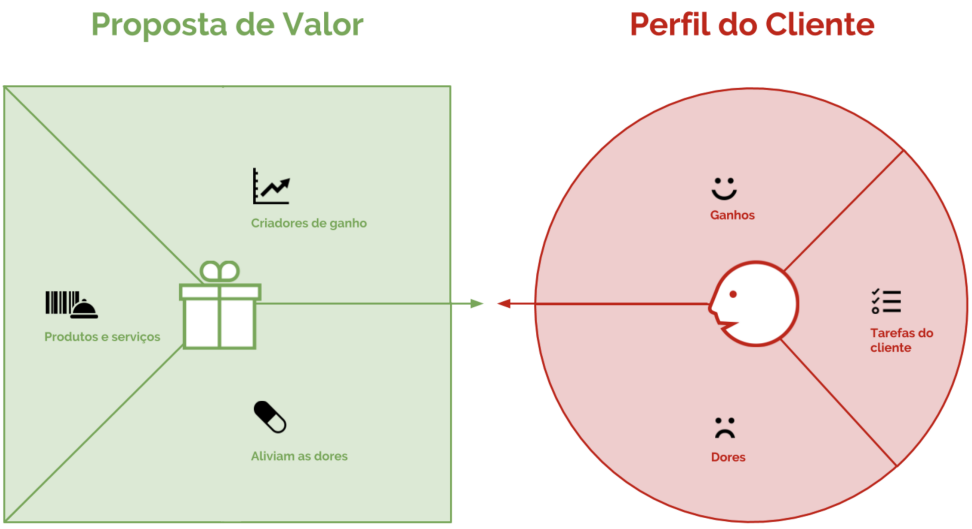
2.3 Ficha de Osterwalder

O diagrama de Proposta de Valor de Osterwalder (OSTERWALDER et al., 2014), também conhecido como Ficha de Osterwalder) é uma ferramenta usada para realizar a correspondência entre produtos oferecidos por empresas e negócios com as necessidades de seu público alvo de maneira estruturada. O diagrama é composto por duas seções: uma focada no cliente e a outra no produto oferecido. Cada seção é subdividida em 3 partes, e a boa conexão entre as partes equivalentes de cada seção (Tarefas do cliente + Produtos e serviços, Ganhos + Criadores de Ganho, Dores + Alívio de Dores) é o que irá determinar se o produto pode atingir um *product-market fit*, isto é, se o produto satisfaz alguma demanda do mercado. A Figura 4 ilustra a Ficha de Osterwalder.

Na seção do cliente, o primeiro ponto que se deve analisar são as tarefas que o cliente deseja realizar. As tarefas representam qualquer tipo de objetivo que o cliente pode ter, incluindo tarefas de cunho funcional (algo a ser realizado ou um problema a ser solucionado), cunho social (algo que promove o *status* da pessoa), cunho emocional (lazer, ter paz de espírito), entre outros. Em seguida, deve se analisar as "dores"(complicações) que o cliente tem ao realizar as tarefas anteriores. Exemplos incluem altos custos monetários, complexidade para realização das tarefas, incertezas do tempo necessário, soluções atuais que são insuficientes, ansiedade, etc. Finalmente, se analisa os ganhos do cliente, que são os benefícios que se obtém ao finalizar as tarefas, como economia de tempo, redução de estresse, impactos sociais, e assim por diante.

Na seção do produto, para se analisar cada uma de suas partes, deve se olhar para cada parte análoga que se encontra na seção do cliente. A parte de produtos e serviços se conecta com as tarefas do cliente, e serve para levantar questionamentos do tipo "Como o meu produto ajuda meu cliente a completar suas tarefas?". A parte de alívio de dores se refere a como o produto irá sanar as dores que o cliente tem, idealmente priorizando as dores que ocorrem com maior frequência ou maior magnitude. Por fim, a parte de

Figura 4: Ficha de Osterwalder



Fonte: (OSTERWALDER et al., 2014)

criadores de ganho se refere a como seu produto irá gerar ou amplificar os ganhos que os clientes esperam ter ao completar suas tarefas.

2.4 Arquitetura - RF, RNF, ODP (Visões)

Todo software é uma tentativa de encontrar uma solução a algum problema (se esse problema é algo que usuários estão dispostos a pagar por uma solução é outro assunto). A maneira que as diferentes partes de um software são estruturadas e conectadas é o que se chama de arquitetura.

Assim como qualquer outro projeto de engenharia, o engenheiro deve projetar a arquitetura do software de acordo com os requisitos e os *tradeoffs* existentes entre as diferentes alternativas: uma escolha correta melhora a eficiência do projeto como um todo, tanto em termos de eficiência computacional, como também na velocidade de desenvolvimento; uma escolha incorreta pode ter reflexos que só serão sentidos muito depois, quando será tarde demais para fazer alterações de maneira ágil.

A seção seguinte falará sobre RM-ODP, que é um padrão que traz diretivas úteis para extrair os requisitos de um projeto e modelar a arquitetura da solução buscada.

2.4.1 RM-ODP (Visões)

O *Reference Model of Open Distributed Processing* (RM-ODP) é um modelo de referência para descrever e padronizar sistemas de computação distribuídos. Esse modelo teve sua origem no esforço conjunto de 3 grandes organizações: a Organização Internacional de Normalização (ISO), a Comissão Eletrotécnica Internacional (IEC) e o Setor de Normatização das Telecomunicações (ITU-T), e está descrito oficialmente na ISO 10746. O RM-ODP aborda vários conceitos diferentes que são necessários para cobrir os mais diversos casos de uso de processamento distribuído como a modelagem de sistemas distribuídos como objetos, um *framework* para testar e detectar problemas em sistemas distribuídos, e a segmentação do sistema em diferentes visões de requisitos. O ponto de maior interesse para esse trabalho é a divisão de um problema em diferentes visões.

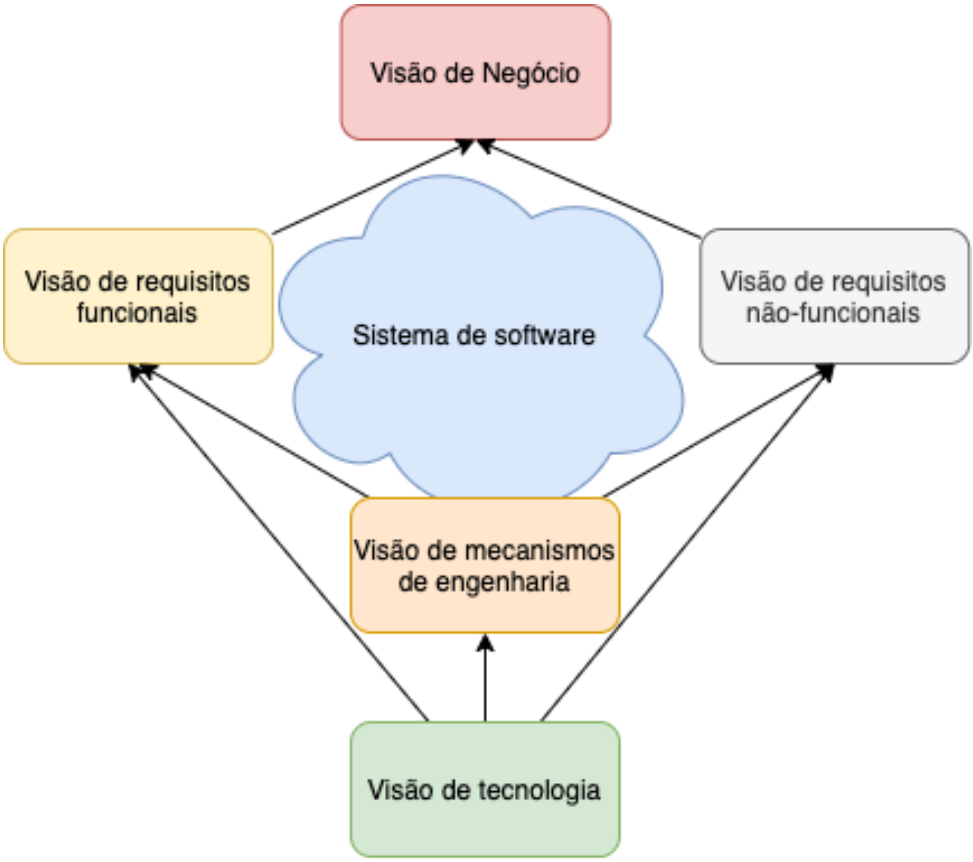
Quando se projeta um sistema, é comum utilizar a técnica de abstração para remover detalhes que não serão pertinente da análise sendo realizada no momento. Tal técnica de olhar para o mesmo componente com diferentes interesses pode ser entendido como novos "pontos de vistas", e é utilizada para gerenciar a complexidade inerente de certas soluções. No esquema RM-ODP, um ponto de vista sempre tem duas partes: os envolvidos e seus interesses. Um envolvido é uma pessoa ou entidade que será afetada pelo sistema sendo projetado. A maneira que os envolvidos utilizam o sistema, seus objetivos, suas expectativas, suas limitações, e suas experiências pessoais, tudo isso são os interesses que os envolvidos tem sobre o sistema, e no final, serão traduzidos para um conjunto de requisitos a serem atendidos. O modelo RM-ODP define 5 pontos de vistas, e a Figura 5 ilustra as relações e dependências entre cada visão:

Visão de negócio: Foca nas necessidades de uma empresa, define o problema, o escopo do projeto, os recursos que serão alocados e os prazos. Por exemplo, imagine que uma empresa gostaria de criar um produto para processar arquivos em larga escala.

Visão de requisitos funcionais: Foca no que o software deve fazer, quais casos de uso que serão implementados e priorizados. Seguindo o exemplo anterior, um requisito de tal sistema poderia ser que ele deve ser capaz de ingerir arquivos com formato CSV, Parquet e Avro.

Visão de requisitos não-funcionais Foca no como o software deve performar durante sua execução. Por exemplo, arquivos menores que 1GB devem ser ingeridos em menos de 1 segundo, arquivos maiores que isso podem demorar até uma hora.

Figura 5: As 5 visões do modelo RM-ODP e suas interdependências.



Fonte: Autoria própria

Visão de mecanismos de engenharia: Foca na interação entre os diferentes componentes do sistema, determina mecanismos para cumprir os requisitos das visões anteriores. Seguindo os exemplos anteriores, poderia se usar uma fila de tarefas com *workers* para permitir que o sistema escale conforme a demanda cresça. Note que a complexidade introduzida por um sistema de filas não é de interesse direto para o negócio, mas é necessário para cumprir os requisitos não-funcionais.

Visão de tecnologia: Foca na implementação concreta dos mecanismos especificados pela visão de engenharia. Por exemplo, como foi determinado que se usaria um sistema de filas, poderia se escolher um produto de fila de mensagens como o RabbitMQ.

2.5 Protocolos de comunicação

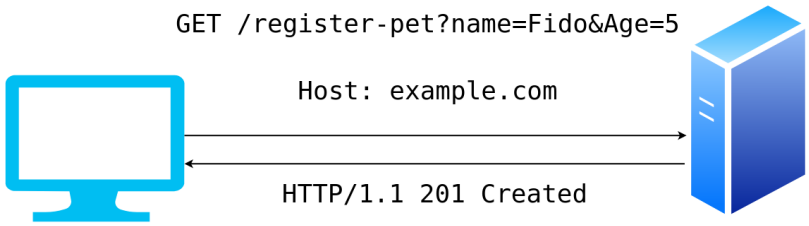
Para dispositivos IoT se comunicarem (entre si ou com uma servidor de aplicação), é preciso que mensagens sejam enviadas de um recipiente para um destinatário via internet.

No modelo OSI (Open System Interconnection), que separa redes em 7 camadas de acordo com suas funcionalidades, dispositivos IoT apresentam certos desafios e requisitos (como necessidade de baixo consumo de banda e custo computacional para processamento) que originaram novos protocolos competidores na camada de aplicação, cada um com suas próprias vantagens e *tradeoffs* por projeto. Essa seção irá descrever brevemente alguns deles no contexto de uso em dispositivos IoT.

2.5.1 HTTP

Um dos protocolos da camada de aplicação mais conhecido é o HTTP (Hypertext Transfer Protocol). É um protocolo cliente-servidor *stateless* e síncrono, em outras palavras, toda informação para processar uma requisição HTTP é auto-contida e não depende de requisições anteriores (*stateless*), e cada cliente espera que sua requisição retorne uma resposta (síncrono). A Figura 6 demonstra um exemplo de uma requisição HTTP sendo enviada de um cliente até um servidor Web. Como é um protocolo já usado em larga

Figura 6: Cliente (esquerda) realizando uma requisição HTTP a um servidor web (direita).



Fonte: Autoria própria

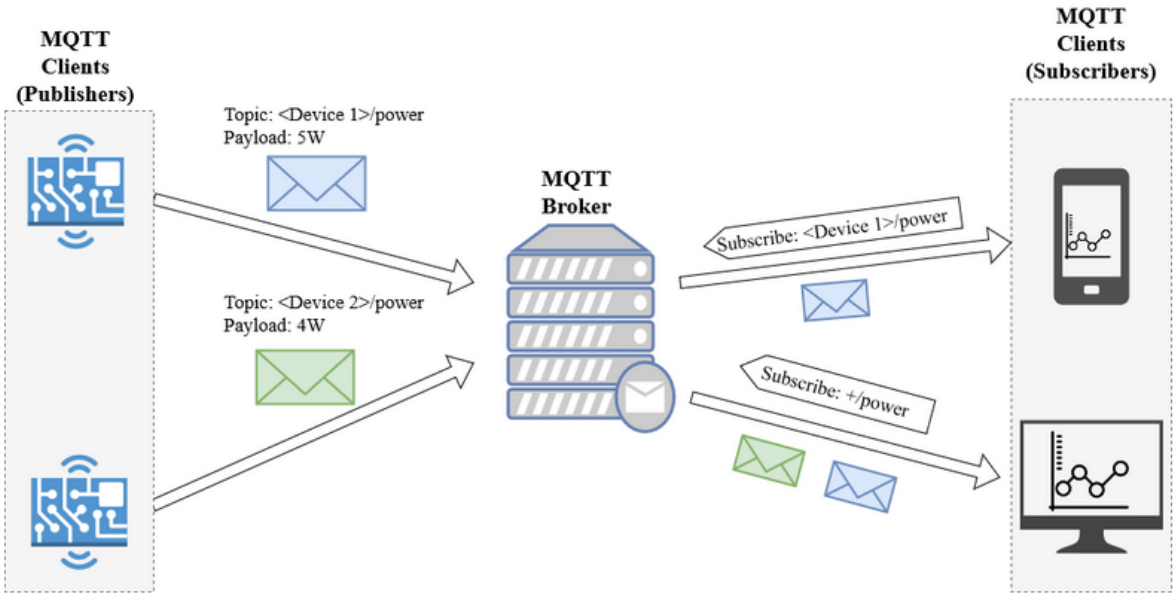
escala no mundo inteiro em servidores Web, uma das grandes vantagens do ponto de vista do arquiteto de um projeto IoT é o número de componentes de hardware e bibliotecas de programas disponíveis para uso imediato.

2.5.2 MQTT

O protocolo MQTT é um protocolo *publish-subscribe stateful* e assíncrono que visa ter baixo consumo energético e baixo uso de largura de banda. Diferentemente da arquitetura cliente-servidor onde clientes se comunicam diretamente com um servidor de aplicação, na arquitetura *publish-subscribe* todas mensagens passam por um *broker* que então a replica e distribui para os *subscribers* (assinantes) interessados em um tópico. A Figura 7 demonstra um *broker* MQTT realizando o *fan-out* de mensagens originadas do dispositivo 1, e também seletivamente distribuindo os mensagens do dispositivo 2 apenas para os

subscribers interessados. Dessa forma, o protocolo é completamente assíncrono, pois o *publisher* nunca espera uma resposta imediata dos *subscribers*. Assim, mesmo que um *subscribers* perca temporariamente conexão com o *broker*, todas mensagens que ele haveria perdido em um protocolo síncrono serão eventualmente recebidas.

Figura 7: *Publishers* (esquerda) publicando mensagens no tópico <id>/power a um *broker* (meio), com *subscribers* (direita) consumindo apenas as mensagens que os interessam.



Fonte: Autoria própria

2.6 Bancos de dados

Uma vez que as mensagens emitidas por dispositivos IoT são recebidas por um servidor de aplicação, caso seja necessário persistir o conteúdo das mensagens, é preciso o uso de um banco de dados. Atualmente existem bancos de dados podem ser classificados em 2 grandes categorias:

2.6.1 Relacionais (SQL)

Bancos de dados relacionais organizam seus dados de forma tabular, com linhas e colunas, e utilizam a linguagem SQL (Structured Query Language) para realização de *queries*. Bancos relacionais são os bancos mais comuns e desde a da década de 80, e possuem um grupo de garantias dentro de transações conhecidas como ACID (HAERDER; REUTER, 1983):

a) Atomicidade

Propriedade que garante que operações dentro de uma transação funcionem ou falhem como um todo; nunca deve ser possível que operações funcionem parcialmente, em caso de falha qualquer resquício da operação deve ser revertido.

b) Consistência

Propriedade que garante que o banco sempre se encontra em um estado válido de acordo com as regras definidas por *schemas*, *triggers*, *cascades*, etc.

c) Isolamento

Propriedade que garante que efeitos gerados por uma transação não podem ser vistos por outras transações até que seja finalizada. Essa condição pode ser relaxada de acordo com o nível de isolamento desejado, porém existe um *tradeoff* entre isolamento e desempenho, quanto maior o nível de isolamento (o maior sendo SERIALIZABLE, onde todas transações são executadas em sequência ao invés de permitir um paralelismo), menor o desempenho.

d) Durabilidade

Propriedade que garante que uma vez que uma transação é finalizada, os resultados da transação devem sobreviver qualquer tipo de falha, como falta de energia elétrica ou uma falha de softwares adjacentes, como um pânico no *kernel* do sistema operacional.

2.6.2 Não-Relacionais (NoSQL)

Bancos de dados não-relacionais organizam seus dados em documentos ou pares chave-valor de forma desnormalizada. Por volta de 2008, com o número crescente de acessos simultâneos em sites de escala global aumentando, empresas buscaram encontrar soluções de bancos de dados que permitissem uma disponibilidade maior do que as permitidas por bancos relacionais comuns. Uma das maneiras de se aumentar a disponibilidade de um sistema é através do uso de escalabilidade horizontal, o que torna o banco de dados um sistema distribuído. O Teorema CAP, introduzido ao mundo por Brewer em (BREWER, 2000), diz que no evento de uma partição de rede (P), não é possível que um sistema distribuído seja ambos consistente (C) e disponível (A, do inglês *available*). Com essa restrição em mente, arquitetos de bancos de dados NoSQL, criaram bancos do tipo BASE (*Basically available, soft state, eventually consistent*), em que se sacrifica a consistência

do Teorema CAP em nome de disponibilidade, e não faz nenhuma garantia transacional ACID como bancos relacionais.

2.6.3 Bancos de dados temporais

Uma série temporal é uma série de pontos ordenados no tempo, onde uma mesma métrica é medida em períodos de tempo discreto. Enquanto bancos comuns lidam muito bem com relações entre entidades, eles deixam a desejar quanto à analise da evolução temporal dos dados que armazena, e por esse motivo, foram inventados bancos de dados especializados. Como séries temporais costumam ser métricas em que grande parte das escritas ocorrem de forma sequencial ao longo do tempo, e cuja granularidade costuma ser mais importante apenas nos dados mais recentes, várias otimizações podem ser empregadas por bancos de dados temporais, por exemplo, *queries* como o cálculo da sazonalidade das vendas de um produto ao longo dos últimos 3 anos podem ser otimizadas de maneira mais eficiente; redução do custo de armazenamento pode ser obtido realizando *downsample* de dados de mais de 3 anos atrás; ingestão online de dados temporais apresenta maior desempenho já que sabemos que escritas geralmente ocorrerão apenas em blocos de tempo recente; funções específicas para análises temporais são implementadas diretamente no banco para maior desempenho, entre outros.

2.7 Web API

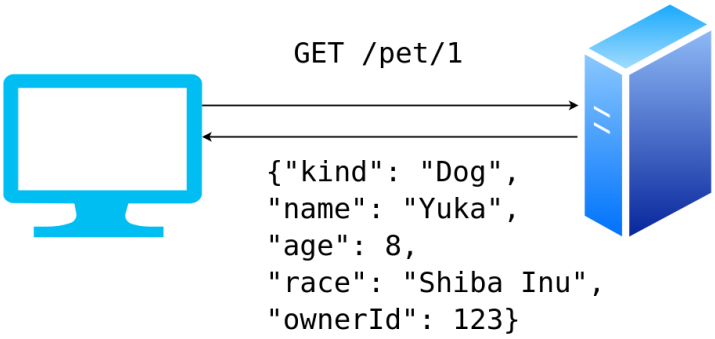
Com os dados dos dispositivos IoT persistidos, é necessário disponibilizá-los de alguma maneira ao mundo externo. Uma maneira comum de se fazer isso é via uma API (Application Programming Interface) que possa ser consumida por outros serviços e clientes. Existem inúmeras maneiras de se organizar uma API, mas atualmente 3 se destacam:

2.7.1 REST

REST (Representational State Transfer) é um estilo de API criado por Fielding em (FIELDING, 2000). Fielding descreve REST como um estilo arquitetural gerado a partir de um conjunto de restrições, que são (a) Arquitetura Cliente-Servidor, (b) Comunicação Sem Estado, onde toda requisição contêm todo o contexto necessário para seu processamento, e o estado é guardado no cliente, (c) Cacheabilidade, toda interação deve ser implicitamente ou explicitamente listada como cacheável ou não-cacheável, (d) Interface Uniforme, onde a maneira que se interage com os recursos é uniformizada, (e) Sistema em

Camadas, promove encapsulamento de outros níveis do sistema, (f) Código em Demanda, o cliente deve ser capaz de baixar funcionalidades extras e códigos executáveis como Java applets e Javascript.

Figura 8: Cliente acessando recurso Pet com ID 1 com resposta serializada em JSON de uma API REST.



Fonte: Autoria própria

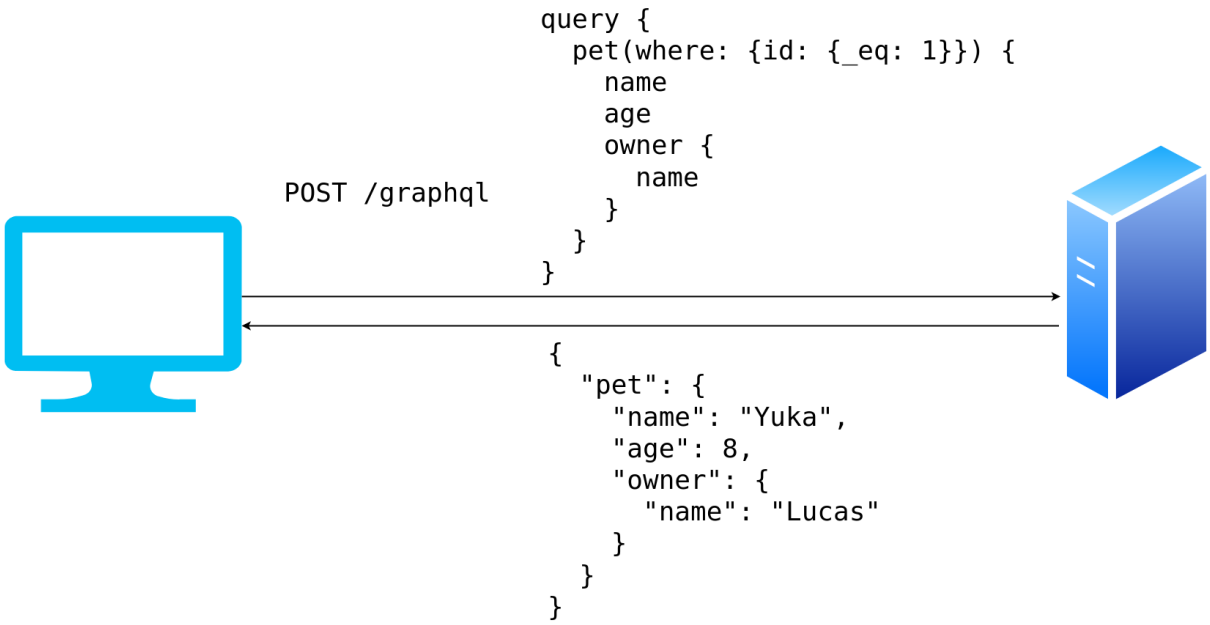
Geralmente utiliza JSON como formato de serialização, mas o padrão arquitetural em si não determina nenhum formato específico. Diferentemente de seu predecessor (SOAP), APIs REST não costumam apresentar um *schema* rígido (apesar de existirem iniciativas como OpenAPI e JSON Schema).

2.7.2 GraphQL

GraphQL é uma linguagem de *query* para APIs. Diferentemente de REST onde servidores especificam o tipo de dado disponível em cada *endpoint*, APIs GraphQL dão maior controle para o cliente requisitar apenas o que desejar ((TAELMAN; SANDE; VERBORGH, 2018)). Desde sua introdução ao mundo pelo Facebook em 2016, a adoção da tecnologia vem crescendo, como demonstrado por empresas como Facebook, GitHub, Twitter, e Yelp. O GraphQL resolve alguns problemas encontrados criados por APIs REST como o *overfetching*, onde o servidor responde com mais informações do que o cliente precisa, e o *underfetching*, onde o servidor não retorna informações suficientes em uma única requisição, e força o cliente a realizar outras requisições para obter informações extras.

A maneira que isso é solucionado em GraphQL é permitindo o cliente pedir exatamente o que deseja saber, e também acessar campos relacionados de uma entidade (geralmente análogo a uma relação em um banco de dados), como por exemplo, acessar simultaneamente informações de um dispositivo IoT e seu dono em uma única requisição.

Figura 9: Cliente acessando campos específicos do recurso Pet com ID 1, juntamente com o campo relacionado "owner"via GraphQL .



Fonte: Autoria própria

Curiosamente, durante o mesmo período em que o GraphQL se tornou *open-source*, outras empresas como Coursera e o Netflix estavam criando suas próprias soluções para os problemas listados acima, mas optaram por adotar o GraphQL.

3 POLILAB - SISTEMA DE GERENCIAMENTO DE DISPOSITIVOS IOT PARA LABORATÓRIOS EDUCACIONAIS

Esta seção irá demonstrar os métodos utilizados para sair da definição do problema e como se chegou na arquitetura de uma possível solução, e finalmente o passo-a-passo da implementação final.

3.1 Contextualização e definição do problema

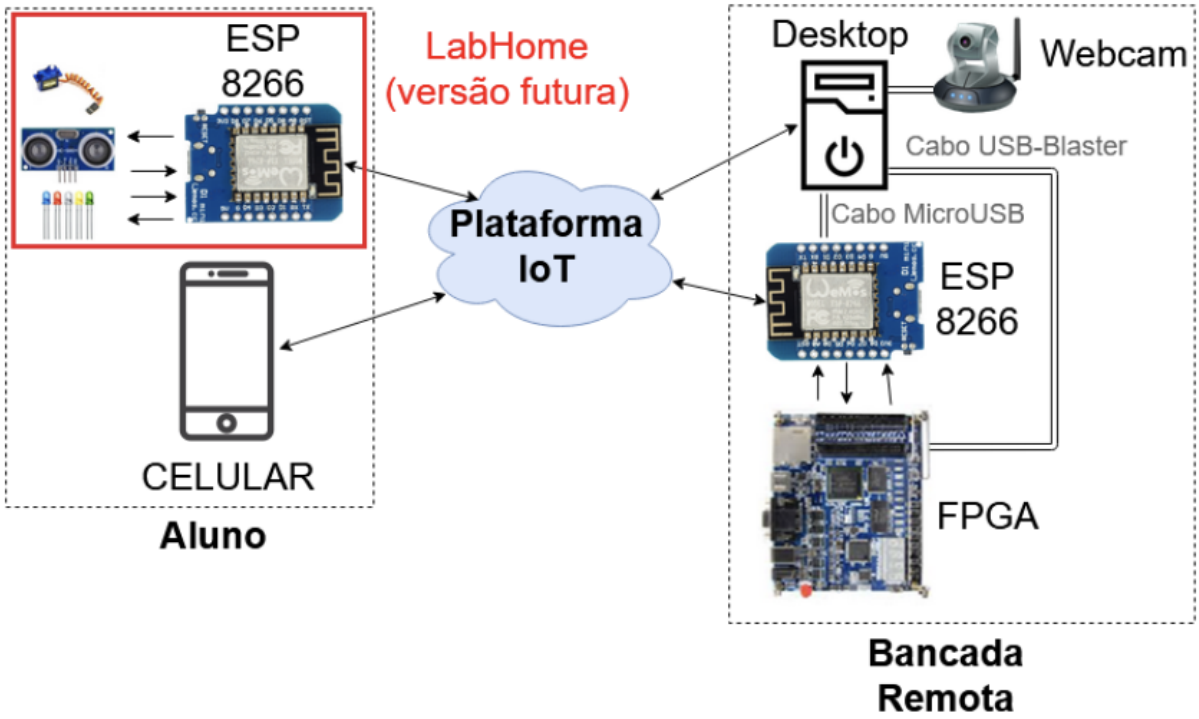
O Laboratório Digital da Poli é um laboratório educacional focado no desenvolvimento de projetos de circuitos digitais (USP, 2021), utilizando componentes discretos e placas FPGA (*Field-programmable gate array*). Alunos inscritos na disciplina realizam experimentos diferentes a cada semana, explorando componentes de hardware diferentes (servo motores, sensores ultrassônicos, comunicação serial) e técnicas como separação do fluxo de dados e do fluxo de controle, e a modelagem de circuitos como máquinas de estados.

Devido a pandemia do COVID-19, surgiu uma necessidade de se disponibilizar métodos de aprendizado remoto para disciplinas da USP. Para disciplinas mais teóricas, a transição pode ser facilmente realizada utilizando programas de videochamadas ou aulas gravadas, porém o mesmo não pode ser feito para disciplinas práticas, principalmente aquelas que requerem equipamentos especializados como é o caso do Laboratório Digital da Poli com suas placas FPGAs. Como geralmente tais equipamentos especializados não possuem nenhuma capacidade de controle remoto via Internet, os funcionários da faculdade acabam se expondo a riscos desnecessários para operá-los fisicamente nos laboratórios.

Dentro desse contexto, em 2020, se iniciou o projeto LabEAD (HAYASHI; HAYASHI; ARAKAKI, 2020), onde foi implementado uma bancada eletrônica de baixo custo para realização de experimentos no Laboratório Digital da Poli de maneira remota. Este projeto

utilizou um conjunto de Arduínos e ESP8266s (microcontrolador Wi-Fi de baixo custo) para realizar o controle das placas FPGA, câmeras para se ter o *feedback* em forma de vídeo (Figura 10). O projeto LabEAD obteve resultados positivos na percepção de docentes e discentes (ALMEIDA et al., 2021), mas introduziu certas complexidades ao ambiente: onde anteriormente existia apenas uma placa FPGA e um Desktop comum, foi necessário introduzir uma frota de dispositivos IoT conectados a uma plataforma na nuvem. Essa complexidade dificulta o gerenciamento do laboratório pelos funcionários, atividades como a configuração dos dispositivos, a programação de horários de operação e o monitoramento do bom-funcionamento de cada um dos equipamentos passam a ser não-triviais.

Figura 10: Arquitetura do LabEAD.



Fonte: (ALMEIDA et al., 2021)

Outro problema é a falta de observabilidade que ambientes de aprendizado remoto acabam promovendo: quando uma matéria é ministrada presencialmente, é relativamente direto para docentes dizerem quem são os (grupos de) alunos que estão se empenhando e interagindo com os equipamentos disponibilizados. Na falta dessa observabilidade, muitas vezes os professores acabam emitindo um maior número de atividades para terem maior visibilidade, sujeitando trabalho extra tanto para os alunos, como aqueles que terão de corrigir as atividades.

Por esses motivos, o projeto desenvolvido nesse trabalho, o PoliLab, começou a ser

desenvolvido. Ao final da implementação, o Laboratório Digital terá acesso a um sistema para gerenciar os dispositivos IoT instalados no local.

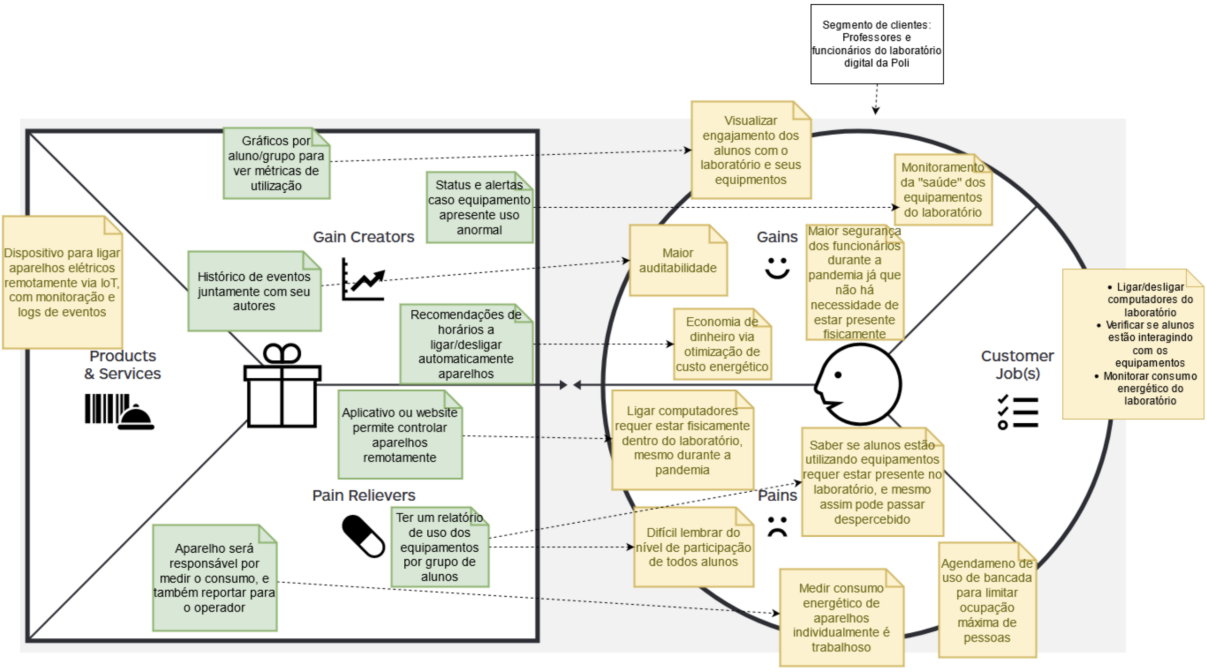
3.2 Ficha de Osterwalder

A Figura 11 foi obtida aplicando os princípios da Ficha de Osterwalder ao projeto deste trabalho, e validado por funcionários do Laboratório Digital da Poli. Apesar do projeto não ter nenhum intuito comercial, é interessante aplicar técnicas desse tipo para averiguar que existe um público alvo interessado e com necessidades não atendidas.

O público alvo deste projeto são os funcionários do Laboratório Digital da Poli, e as principais tarefas identificadas foram ligar e desligar os computadores e placas FPGA do laboratório, ter um monitoramento mais próximo do engajamento dos alunos com equipamentos, e também ter um monitoramento maior do consumo energético do laboratório.

A partir dessa ficha, fica evidenciado que a solução proposta trará benefícios a um público alvo existente, e serve de ponto de partida para a modelagem da arquitetura do projeto.

Figura 11: Ficha de Osterwalder do projeto implementado neste trabalho.



Fonte: Autoria própria

3.3 Visões Arquiteturais - RM-ODP

Apesar da Ficha de Osterwalder não ter uma relação direta com o modelo RM-ODP, há uma progressão natural dela para a Visão de negócio.

3.3.1 Visão de negócio

A definição do produto obtido a partir da Ficha de Osterwalder especifica que será necessário criar um dispositivo IoT para controlar remotamente qualquer tipo de aparelho do laboratório. Dessa definição, é possível dividir o projeto em duas partes principais, sendo elas:

- Projeto do hardware do dispositivo IoT a ser colocado dentro do laboratório
- Projeto do sistema de controle dos dispositivos IoT via Internet

Como cada um desses projetos tem escopo grande o suficiente para ser sua própria tese, este trabalho focará apenas no projeto do sistema de controle, e utilizará o projeto de hardware baseado nos trabalhos de (HAYASHI; ALMEIDA; KOMO, 2021) e (HAYASHI; HAYASHI; ARAKAKI, 2020).

Os principais casos de uso do identificados no produto foram os seguintes:

- Como leitor, quero poder acessar métricas públicas dos laboratórios.
- Como leitor, não tenho permissão para realizar ou agendar operações, e nem acessar dados privados.
- Como um funcionário, tenho permissão para fazer tudo que um leitor pode fazer.
- Como um funcionário, quero poder ligar aparelhos remotamente.
- Como um funcionário, quero poder desligar aparelhos remotamente.
- Como um funcionário, quero poder agendar operações em horários e dias específicos para equipamentos específicos.
- Como um funcionário, quero ser alertado o mais rápido possível caso haja algum problema com os equipamentos.
- Como um funcionário, quero visualizar métricas relacionadas ao engajamento de alunos.

- Como um funcionário, quero visualizar métricas relacionadas ao consumo energético do laboratório.
- Como um administrador, posso fazer tudo que um funcionário comum faz.
- Como um administrador do sistema, quero poder adicionar novos membros funcionários ao laboratório.
- Como um administrador do sistema, quero poder registrar novos dispositivos IoT a serem incorporados no laboratório.
- Como o dono do laboratório, posso fazer tudo que um administrador faz.
- Como o dono do laboratório, quero ter certeza que apenas eu posso realizar operações destrutivas (deletar recursos).

3.3.2 Visão de requisitos funcionais

A partir dos casos de uso que a visão de negócio estabelece, é possível extrair uma séries de requisitos funcionais, entre eles:

RF01 - Disponibilizar métricas do laboratório para leitura por usuários autenticados. Métricas de consumo energético devem ser acessíveis por qualquer leitor, mas métricas relacionadas a alunos devem ter acesso restrito.

RF02 - Controlar aparelhos remotamente para cumprir o objetivo primário do projeto de permitir controle remoto dos equipamentos dos laboratórios educacionais. As operações mais primárias seriam ligar e desligar os equipamentos.

RF03 - Agendar operações para um grupo arbitrário de aparelhos para que o gerenciamento dos equipamentos possa ser feita de maneira estruturada e planejada com antecedência.

RF04 - Alertar funcionários caso haja alguma anomalia para que ações corretivas possam ser realizadas.

RF05 - Autenticar usuários para garantir que o usuário realmente é quem ele afirma ser.

RF06 - Autorização de operações para que apenas usuários com as devidas permissões possam realizar funções restritas.

3.3.3 Visão de requisitos não-funcionais

A partir dos casos de uso que a visão de negócio estabelece, é possível extrair uma séries de requisitos não-funcionais, entre eles:

RNF01 - Autorização deve ter um caráter hierárquico para que usuários mais privilegiados como administradores e donos de laboratórios possam realizar todas operações que usuários abaixo deles tem permissão.

RNF02 - Métricas devem ser coletadas com uma latência menor que 10 segundos para que as ações reativas (como desligar um aparelho que foi ativado acidentalmente) possam ser realizadas o mais cedo possível.

RNF03 - Comunicação deve ser tolerante a falha, mais especificamente, a troca de mensagens entre o sistema de controle e os dispositivos IoT. Em caso de falhas, a mensagem enviada por um dos lados deve eventualmente ser entregue ao destinatário pretendido pelo menos uma vez.

RNF04 - Banda consumida por mensagem deve ser menor que 500B para que a aplicação tenha o menor impacto possível na banda da rede do laboratório.

3.3.4 Visão de mecanismos de engenharia

Com os requisitos estabelecidos, pode se começar o design dos mecanismos de engenharia. Os requisitos estabelecem a necessidade de 5 subsistemas, que serão descritos nas subseções seguintes. A Tabela 3 especifica quais requisitos são cumpridos por qual subsistema.

3.3.4.1 Subsistema de autenticação

O subsistema de autenticação é responsável por validar a identidade de um usuário específico. Existem múltiplos fatores de autenticação disponíveis:

Aquilo que o usuário sabe É o fator mais comum encontrado em aplicações, consiste em pedir um segredo (senha) que apenas o usuário saberia. É um método de autenticação simples de se implementar, porém é comum que usuários tenham senhas fracas ou que vazem suas senhas.

Tabela 3: Mapeamento de requisitos funcionais e não-funcionais com o sistema que os implementa

Subsistema	**Requisitos**
Autenticação	RF05
Autorização	RF06
	RNF01
Comandos	RF02
	RNF03
	RNF04
Agendamento	RF03
Monitoramento	RF01
	RF04
	RNF02
	RNF03
	RNF04

Fonte: Autoria própria

Aquilo que o usuário têm É análogo a um sistema de chave e fechadura. É mais resiliente que uma senha pois não é algo que pode ser inferido ou descoberto, e esquemas criptográficos modernos basicamente impossibilitam a falsificação dessas chaves. Dificilmente um atacante terá acesso a um objeto pessoal do usuário como seu celular ou seu crachá.

Aquilo que o usuário é Também conhecido como biometria. Não é muito recomendado pois diferentemente de senhas, cadastros biométricos não são revocáveis, e já existe registros de ataques que utilizam fotos de alta resolução para reconstruir digitais.

Autenticação é relevante em qualquer aplicação onde se tem conteúdo restritos a um grupo de usuários, e no caso desse projeto, seria esse conteúdo seria o acesso a dados privados de cada laboratório e a autorização para envio de comandos. Um esquema de senhas será suficiente para atender os requisitos estabelecidos pelas visões anteriores, porém idealmente se utilizaria mais de um fator para se validar a identidade de um usuário.

3.3.4.2 Subsistema de autorização

O subsistema de autorização é responsável por assegurar que os usuários só terão acesso aos conteúdos que lhe foram concedidos permissão. Existem múltiplos esquemas de autorização, sendo os mais comuns:

Access control list (ACL) É um esquema simples onde um recurso está atrelado a

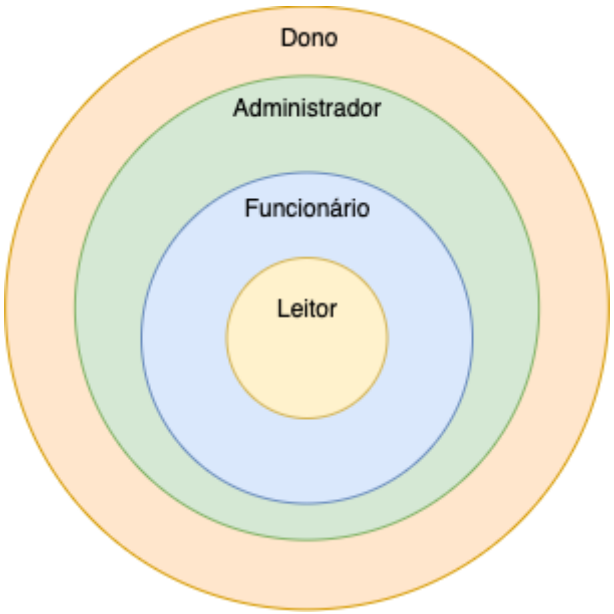
uma lista de usuários que específica se eles tem permissão para utilizar tal recurso. É análogo a um segurança barrar a entrada em um clube baseado numa lista de convidados.

Role based access-control (RBAC) Esquema onde suas permissões estão atrelados a seu cargo dentro do sistema. Pode ser visto como um nível de indireção adicionado a uma ACL. Seguindo a analogia anterior, o segurança agora permite entrada para aqueles que tem um bracelete colorido ao invés de procurar nomes específicos.

Attribute based access-control (ABAC) Esquema onde propriedades do usuário definem se ele tem acesso ao recurso. Seguindo a analogia anterior, o segurança agora permite entrada apens para aqueles que são maiores de idade.

Os requisitos especificam que apenas certos tipos de usuários tem acesso a realizar certas operações. Ao invés das regras de autorização estarem espalhadas em vários locais da aplicação (o que dificulta auditabilidade), seria interessante se todas as regras estivessem centralizadas em um único subsistema. Como os próprios requisitos mencionam cargos (leitor, funcionário, administrador, dono), o uso de RBAC é apropriado.

Figura 12: Hierarquia dos cargos da aplicação



Fonte: Autoria própria

É importante ressaltar que a autorização nessa aplicação tem um caráter hierárquico, de forma que o conjunto de operações disponível para cada cargo é um subconjunto das operações disponíveis para o cargo superior (Figura 12).

3.3.4.3 Subsistema de comandos

O subsistema de comandos é responsável por transmitir os comandos de um usuário para um dispositivo específico, e deve garantir que a mensagem chegue pelo menos uma vez no recipiente. É nesse subsistema que é implementado a lógica para relacionar comandos com seus dispositivos alvos.

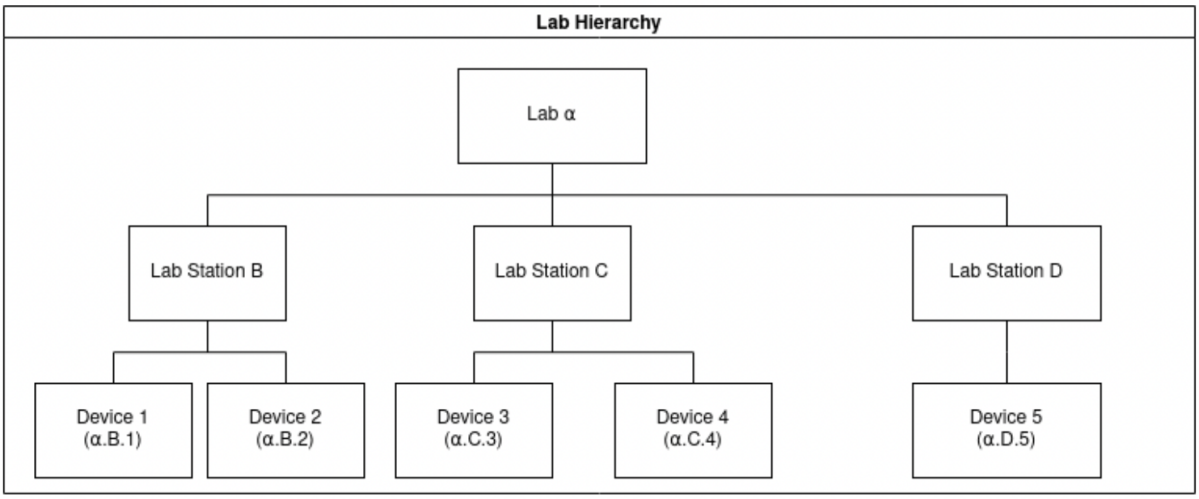
Numa implementação ingênua, durante a criação do comando, se atribuiria manualmente todos os dispositivos que esse comando deve operar. Por exemplo, se quisermos desligar todos os 5 dispositivos do laboratório, digitaria-se os 5 IDs. Isso até funciona, mas não é ergonômico em situações em que se adiciona mais dispositivos ao laboratório. O operador teria que lembrar de adicionar esse novo dispositivo a todas as regras aplicáveis, e a chance de existir erro humano é alta.

Para resolver isso, pode-se adicionar um nível de indireção e desacoplar a definição do comando com seus alvos. Em vez do comando receber uma lista de IDs de dispositivos em que deve ser invocado, ele recebe um conjunto de regras para decidir quais dispositivos agir sobre. Por simplicidade, essas regras podem ser representadas via *tags* que denotam algum atributo do dispositivo. A fim de exemplo, será usado um laboratório fictício com a hierarquia mostrador pela Figura 13. Os dispositivos em si deste laboratório não possuem nenhuma relação lógica entre eles, porém para operadores humanos, é difícil gerenciar uma grande frota de dispositivos individualmente, então é comum se realizar algum agrupamento deste dispositivos de alguma forma. Na hierarquia desse laboratório, esse agrupamento foi feito utilizando o conceito de "bancadas", que por sua vez também são agrupados dentro de um escopo pai chamado "laboratório α".

Suponha então que exista um comando que deve ser invocado apenas nos dispositivos da bancada B. Inicialmente, os dispositivos 1 e 2 foram assinalados diretamente ao comando, e tudo funcionou como esperado. Porém, depois de um tempo, o dispositivo 2 falhou e foi preciso trocá-lo pelo dispositivo 12. Caso não haja nenhuma indireção, o operador teria que analisar todos os comandos registrados no sistema (que podem ser centenas) e trocar todas ocorrências do dispositivo 2 pelo 12. Isso gera uma carga de trabalho manual e sujeita a erros humanos. Caso fosse utilizado um motor de regras, o ato de assinalar o dispositivos 12 à bancada B seria suficiente para que o sistema inferisse quais regras foram afetadas.

O mecanismo de *tags* representando atributos e agrupamentos lógicos de dispositivos é flexível e permite criar agrupamentos arbitrários, permitindo que o subsistema de comandos seja bem versátil.

Figura 13: Hierarquia de um laboratório fictício



Fonte: Autoria própria

3.3.4.4 Subsistema de agendamento

O subsistema de agendamento é responsável por invocar o subsistema de comandos nos horários agendados pelos operadores. Eventos não recorrentes tem um comportamento simples, basta invocá-los no *timestamp* especificado dentro de uma margem de tolerância. Mudanças como horário de verão ou troca de fuso horário podem quebrar expectativas de usuários caso não seja utilizada uma representação apropriada do *timestamp*, por isso será utilizado o modelo da ISO8601.

Eventos recorrentes são mais complexos, e existem 2 métodos principais para definir suas regras de agendamento:

cron é um programa encontrado em linhas de comando de sistemas Unix. Por sua simplicidade de processamento e especificação, o formato utilizado pelo **cron** foi adotado em vários sistemas de agendamento. A fim de exemplo, para representar uma operação que deve ocorrer toda quinta-feira as 15:30 e 18:30, seria utilizado a expressão "30 15,18 * * THU".

iCalendar (RFC 5545) Definido pela *Internet Engineering Task Force* (IETF), é um formato padronizado para permitir que aplicações especifiquem e comuniquem informações de agendamento de eventos. Permite exprimir regras de recorrências (RRULES) complexas que não são representáveis no esquema cron (por exemplo, um evento que ocorre uma vez a cada N dias). Devido a complexidade da especificação, implementações de sistemas de agenda desse tipo acabam sendo mais raras.

Para esse projeto, idealmente se utilizaria um sistema compatível com o formato da RFC5545 para permitir exportar eventos para outros sistemas de calendários como o Google Calendar, mas como os casos de uso não especificam a necessidade de utilizar regras de recorrência complexas, o uso de expressões cron é suficiente. É importante ressaltar que caso houvesse requisitos para a aplicação operar em um ambiente internacional, seria necessário introduzir uma camada de tradução para sistemas de calendários não-gregorianos como os calendários islâmico e chinês, que tem suas próprias peculiaridades.

3.3.4.5 Subsistema de monitoramento

O subsistema de monitoramento é responsável por coletar, agregar e disponibilizar métricas obtidas dos dispositivos IoT. É importante que o sistema receba pelo menos uma vez cada métrica emitida pelos dispositivos, mesmo que o sistema perca conectividade temporariamente. Tal sistema pode se tornar quão complexo quanto necessário com o uso de técnicas como relatórios programados, detecção de anomalias em tempo real, notificação de eventos, entre outros. Esse sistema é necessário pois os efeitos dos comandos invocados pelo subsistema de comandos só são perceptíveis de maneira assíncrona, então é preciso ter um sistema que se inscreva aos eventos e métricas emitidos pelos dispositivos IoT.

3.3.4.6 Arquitetura proposta

A arquitetura proposta para o projeto pode ser sucintamente observada na Figura 14. O sistema de autenticação e autorização agem como uma barreira inicial antes de permitir o acesso aos subsistemas mais internos, que se comunicam livremente entre si.

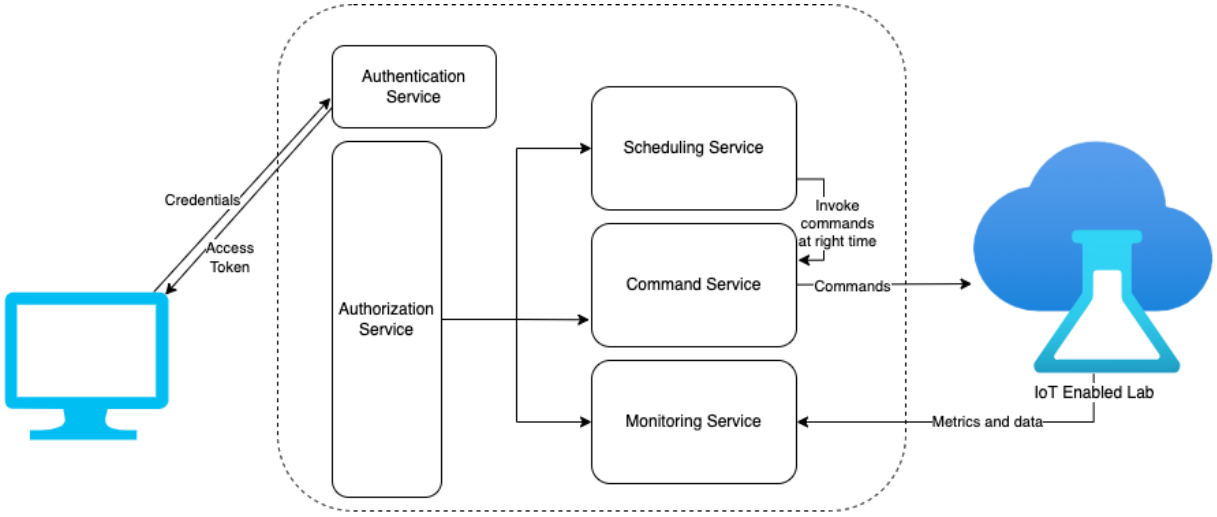
3.3.5 Visão de tecnologia

Com os mecanismos de engenharia definidos, a visão de tecnologia propõe escolhas concretas para a implementação de cada subsistema. No caso de não existir nada no mercado que cumpra os requisitos dentro do orçamento, a implementação deve ser feita pela própria equipe (ou o projeto deve ser repensado).

3.3.5.1 Subsistema de autenticação

O subsistema de autenticação proposta necessita implementar 3 fluxos básicos: cadastro de novos funcionários, login e troca de senha. O Firebase é um produto *Backend As A*

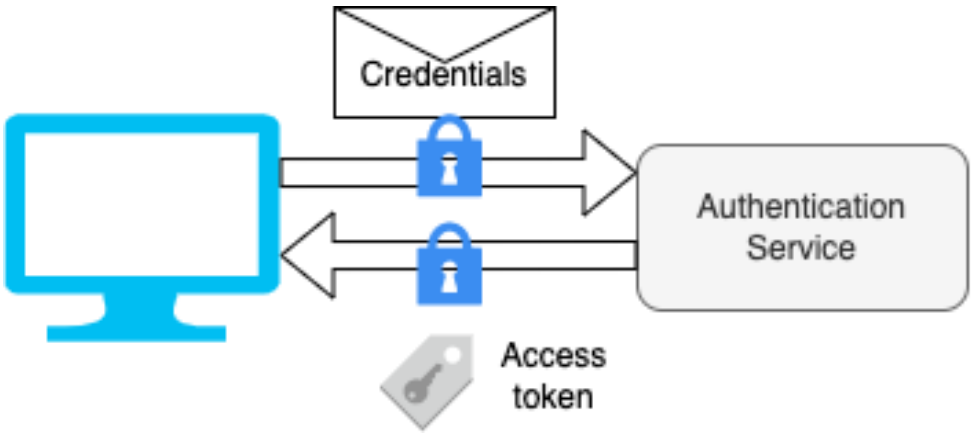
Figura 14: Arquitetura obtida com após divisão em subsistemas



Fonte: Autoria própria

Service do Google, que provém soluções prontas para serviços como autenticação, autorização, armazenamento, *crash analytics* e outros. O Firebase cumpre todos requisitos do projeto, além de fornecer algumas funcionalidades extras como logins sociais via o fluxo OAuth (HARDT et al., 2012), e evita riscos desnecessários relacionados a vazamento de senhas ou uso de métodos criptográficos fracos. No contexto desse projeto, foi utilizado principalmente pela funcionalidade de provedor de autenticação via emissão de tokens JWT (TOKEN, 2015) (JSON Web Token), mas poderia ser facilmente substituído por uma solução customizada ou por outro provedor terceiro como Auth0 ou Okta. A maior vantagem que o Firebase tem sobre os outros provedores é o seu modelo de cobrança: o plano base do Firebase é gratis, mas a Okta cobra 2 dólares por usuário.

Figura 15: Cliente realizando fluxo de login com o subsistema de autenticação



Fonte: Autoria própria

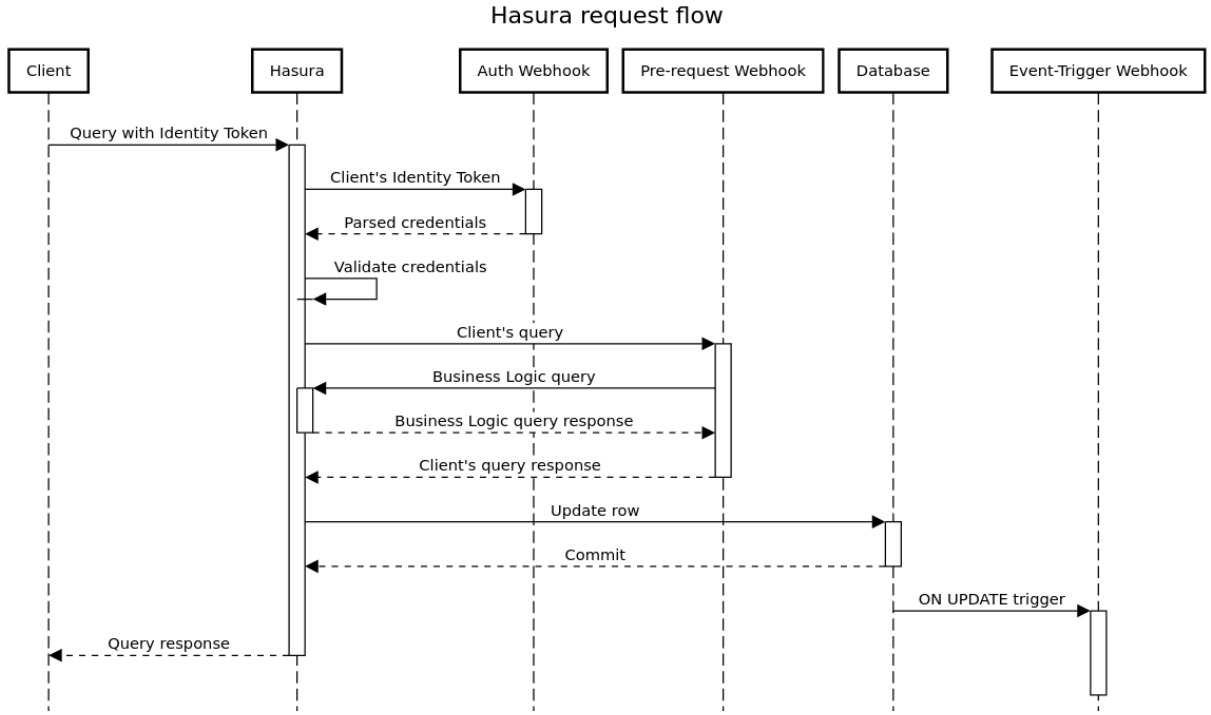
3.3.5.2 Subsistema de autorização

O principal requisito para o subsistema de autorização é de permitir que sejam criadas regras arbitrárias para permitir ou barrar operações dependendo do cargo que o usuário possui dentro do laboratório. Existem produtos dedicados para análise de permissões como o *Open Policy Agent*, mas nesse projeto se optou por utilizar o Hasura já que ele cumpre os requisitos de autorização, e também benefícios adicionais na prototipação rápida de uma API.

O Hasura é um *engine open-source* para conversão de *queries* GraphQL em SQL. Sem nenhuma configuração, o Hasura provê uma API CRUD (Create-Read-Update-Delete, as operações básicas que um banco de dados implementa) completa, mas CRUD apenas não é suficiente para implementar regras de negócio complexas. Com isso em mente, o Hasura foi projetado para ter uma arquitetura extensível, permitindo implementar regras de negócio de forma reativa utilizando *webhooks* (*endpoints* HTTP a serem invocados quando um certo evento ocorre).

O Hasura expõe 3 tipos de *webhooks* principais: *webhooks* de autenticação, *webhooks* pré-requisição, e *webhooks* de *triggers* de banco de dados.

Figura 16: Fluxo de uma requisição dentro do Hasura usando 3 tipos de webhooks.



Fonte: Autoria própria

Como *tokens* de identidade originados de provedores de autenticação podem ter for-

matos diversificados (Cookies, JWT, SAML Response, entre outros), o Hasura depende do *webhook* de autenticação para a tradução de um *payload* arbitrário em um formato conhecido. Já os *webhooks* pré-requisição, conhecidos como *Hasura actions*, são invocados antes do *engine* executar *queries* no banco, e aqui é possível implementar lógica arbitrárias como validação de *inputs* do usuário ou publicação de mensagens MQTT. Finalmente, os *webhooks* de *event-triggers* são invocados quando um certo evento ativa um *trigger* no banco de dados. Por exemplo, quando um dispositivo IoT muda de estado (ligado para desligado, ou vice-versa), a atualização da entrada no banco de dados ativa um *trigger* que invocará um *webhook* para notificar as partes interessadas. O Hasura permite especificar regras de autorização para todas operações via um arquivo de configuração em formato JSON, e essas regras são avaliadas antes do *webhook* pré-requisição ser invocado.

3.3.5.3 Subsistema de comandos

Como o design do subsistema de comandos é algo bem específico ao projeto, não existe nenhuma solução pronta para ser utilizada, então terá que ser implementado do zero. A decisão a ser tomada agora é definir qual linguagem de programação utilizar, já que cada linguagem tem seus próprios prós e contras. Os principais pontos a serem analisados são a facilidade de uso, a riqueza do ecossistema da linguagem, e a velocidade de execução. Com isso em mente, a linguagem escolhida para desenvolvimento do projeto foi o Typescript. O NPM é o repositório de pacotes para bibliotecas de Javascript, e no momento é o maior repositório de pacotes do mundo, com mais de 1.3 milhões de pacotes (3x mais que o Pypi, o principal repositório da linguagem Python). O Typescript é um superconjunto da linguagem Javascript (com benefícios adicionais de análise estática de código), e por isso é compatível com todas suas bibliotecas, o que demonstra a riqueza do ecossistema dessa linguagem. A facilidade de uso é algo subjetivo a cada indivíduo, mas visto que o projeto tem potencial para ser estendido em anos futuros, seria interessante utilizar uma linguagem popular. Como o Typescript é uma das 5 linguagens mais populares (STACK OVERFLOW, 2021), certamente se qualifica. Por fim, como o limitante para garantir que a comunicação entre o sistema de comandos e os dispositivos IoT ocorra dentro de uma janela de tolerância é a latência da rede, não é necessário utilizar uma linguagem de super alta performance, então não há problemas em utilizar linguagens interpretadas.

Para cobrir o requisito de tolerância a falha na comunicação desse subsistema, é interessante o uso do protocolo de comunicação MQTT, que faz garantias sobre as entregas de mensagens. Para tal, decidiu-se utilizar o HiveMQ como implementação do *broker* MQTT, já que se trate de um projeto de código aberto e uso grátis. Outro ponto relevante

```
{
  "_exists": {
    "_where": {
      "_and": [
        {
          "lab_id": {
            "_eq": "X-Hasura-lab-id"
          }
        },
        {
          "user_id": {
            "_eq": "X-Hasura-User-Id"
          }
        },
        {
          "role": {
            "path": {
              "_ancestor": "owner.admin"
            }
          }
        }
      ]
    },
    "_table": {
      "schema": "public",
      "name": "lab_user_role"
    }
  }
}
```

Listing 1: Exemplo de especificação de regra de autorização no Hasura: apenas usuários com cargo de administrador ou superior podem realizar tal operação

para a decisão do uso do Typescript como linguagem para desenvolvimento do projeto é que verificou-se que a biblioteca mais popular e completa para trabalhar com o protocolo MQTT (utilizando a métrica de número de estrelas no repositório) foi escrita para o Javascript.

3.3.5.4 Subsistema de agendamento

Como foi definido que se utilizaria o esquema cron para definir as regras de recorrência de invocamento dos comandos, pode-se utilizar qualquer *scheduler* que implemente tal formato, como o BullMQ, um *daemon* cron, ou até um serviço como o Google Cloud

Scheduler. Porém, visto que já se optou por utilizar o Hasura no sistema de comandos, e razoável utilizá-lo também como o *scheduler* de operações. Convenientemente, o Hasura suporta ambos agendamentos recorrentes e agendamentos de uso único, que cobre ambos casos de uso especificados pela visão de requisitos funcionais.

3.3.5.5 Subsistema de monitoramento

Para realizar o armazenamento das métricas coletadas pelo sistema de monitoramento, é preciso utilizar um banco de dados relacional para facilitar futuras análises, e para tal foi escolhido o PostgresSQL. O PostgreSQL, também conhecido como Postgres, é um banco de dados relacional *open-source* criado pela Universidade da Califórnia em Berkeley. Um dos objetivos originais do projeto foi ser um banco extensível, e por isso costuma ser uma base para empresas buscando entrar no espaço do desenvolvimento de novos bancos de dados. Hoje o Postgres é um dos bancos de dados mais respeitados na indústria, sendo o segundo mais popular atualmente (STACK OVERFLOW, 2021), e é usado em inúmeras empresas de escala global como Instagram, Reddit, Skype, e Yandex.

Além de relações entre as entidades dos laboratórios, é preciso armazenar dados de séries temporais de maneira estruturada, e o TimescaleDB é o complemento para aplicações que já utilizam o Postgres. O TimescaleDB é um produto *open-source* criado pela empresa Timescale que traz capabilidades de bancos de dados temporais ao PostgreSQL. Enquanto que certos competidores no mercado de bancos de dados temporais requerem o uso de uma *query language* especializada (InfluxQL ou Flux no InfluxDB, PromQL no Prometheus) ou de abandonar o modelo relacional completamente, o TimescaleDB mantém compatibilidade com o *SQL standard*, e permite trazer todas suas funcionalidades à um banco Postgres já existente. A maneira que o TimescaleDB faz isso é usando o que eles chamam de *Hypertables*, uma estrutura abstrata que provém uma *view* unificada que pode ser interagida usando comandos SQL, mas que é construída a partir de pedaços independentes das séries temporais, que podem ou não viver em nós separados. Como o Timesc aleDB é apenas uma extensão do PostgreSQL (literalmente instalado via o comando CREATE EXTENSION), é possível criar relações entre entidades do banco Postgres com as métricas gerenciadas pelo TimescaleDB.

3.4 Processo de implementação

O sucesso da implementação de um projeto de software requer, acima de tudo, um bom gerenciamento de tempo, custo, e outros tipos de recursos. Um desses recursos são as expectativas do time, e é crucial que os *stakeholders* e o membros da equipe estejam de acordo com o que será feito. Por isso, a cada 15 dias foram feitas reuniões de alinhamento. Isso ajudou a assegurar que o projeto caminhasse da maneira esperada, e evitou divergências entre a implementação e os planos originais.

Outro gerenciamento necessário (e específico a esse projeto) é o gerenciamento de dispositivos, e para realizar isso o sistema PoliLab precisa se comunicar com algum hardware IoT existente no Laboratório Digital. Para acelerar o desenvolvimento do projeto, se decidiu utilizar o projeto implementado em (HAYASHI; ALMEIDA; KOMO, 2021) para controlar as placas FPGA utilizando mensagens MQTT. A presença desse acelerador permitiu focar todos os esforços no software, e reaproveitar um trabalho bem desenvolvido de hardware. A Figura 17 mostra um exemplo de um dos dispostivos IoT instalados no laboratório.

Figura 17: Foto do hardware do dispositivo IoT instalado no laboratório.

Fonte: (HAYASHI; ALMEIDA; KOMO, 2021)

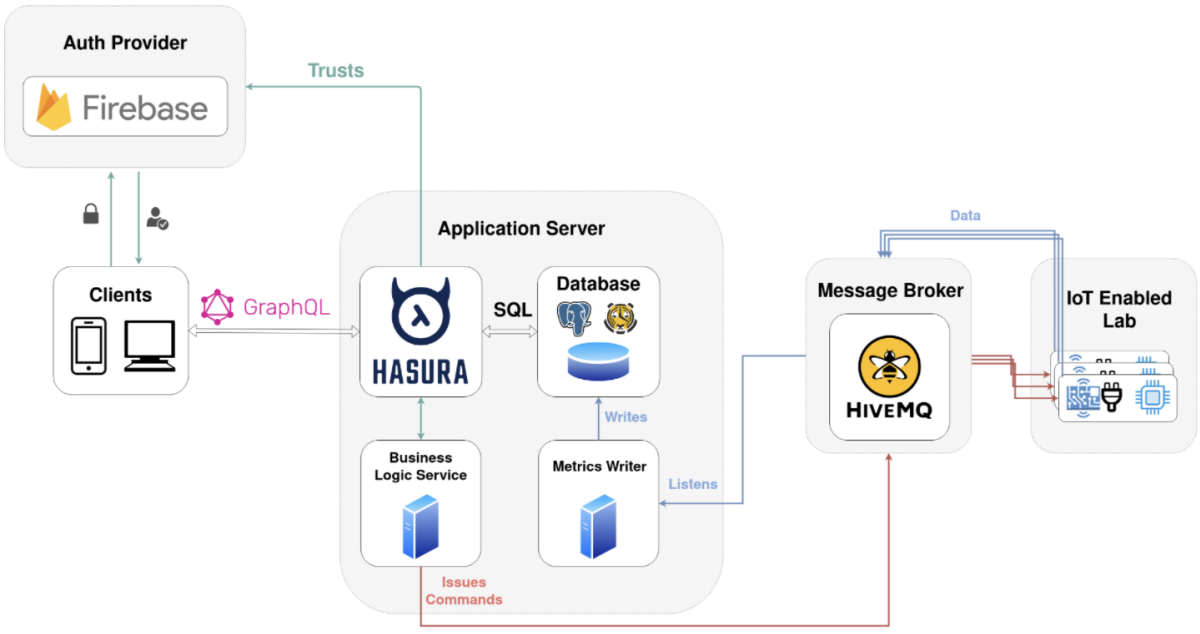
Por serem parte do fluxo crítico da aplicação (enviar comandos e ler dados) os subsistemas de comando e monitoramento foram priorizados. Coincidentemente, ambos sistemas são os mais complexos e requisitam uma implementação customizada, já que não existe

nenhum software de código aberto que cumprisse todas os requisitos. Depois de finalizado ambos sistemas anteriores, foi feita a integração com os sistemas de autenticação e autorização. Como a arquitetura do projeto previu a necessidade desses sistemas separados, e pelo amplo suporte das tecnologias escolhidas, a integração ocorreu sem maiores problemas. O subsistema de agendamento inicialmente foi projetado para utilizar o esquema cron, mas durante o desenvolvimento do projeto, decidiu-se trocar para o esquema da RFC5545 para que se pudesse exportar as agendas de comandos para programas terceiros como o Google Calendar. Graças as reuniões quinzenais, foi possível averiguar que não haveria tempo suficiente para implementar um *scheduler* compatível com a RFC do zero, e evitando raciocínios como o *sunk cost fallacy*, se decidiu que seria necessário reverter a decisão e utilizar o esquema cron como planejado originalmente. Isso representou uma parcela significativa de esforço perdido, mas o sistema foi completado a tempo. Com todos subsistemas finalizados, o projeto pode prosseguir para a parte final, onde foi implementado uma interface gráfica para que os operadores gerenciem o sistema de forma intuitiva. A interface e os modos de uso serão detalhados na próxima seção.

4 RESULTADOS

O objetivo do trabalho foi implementar um sistema para permitir integrar dispositivos IoT instalados em laboratórios e gerenciá-los remotamente. A arquitetura do sistema implementado juntamente com as tecnologias utilizadas podem ser vistas na Figura 18. Como o sistema foi projetado de maneira modular, é possível escalonar o *deployment* de cada uma das caixas do diagrama individualmente, de forma a aumentar a disponibilidade do sistema através de redundâncias via réplicas.

Figura 18: Arquitetura e tecnologias usadas nos subsistema do PoliLab

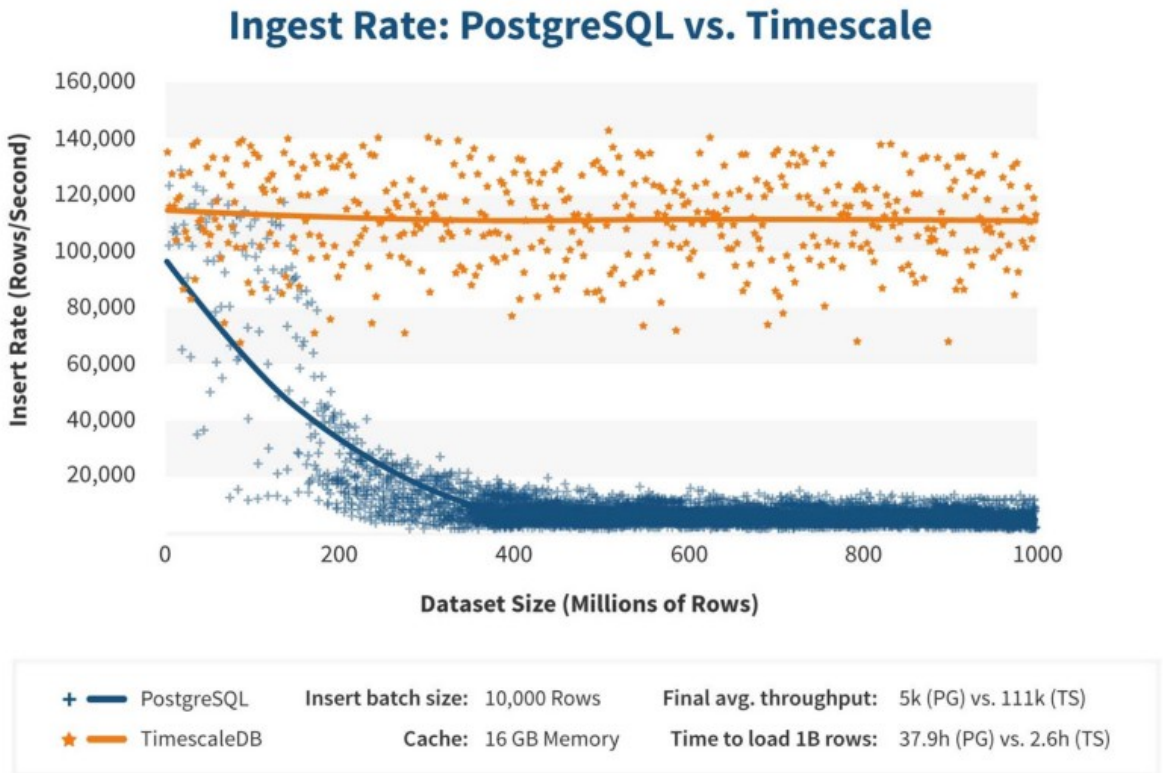


Fonte: Autoria própria

Focando em disponibilidade, é preciso identificar o gargalo do sistema. Como as camadas de aplicação não realizam nenhum processamento intenso de dados, e sabendo que o *broker* MQTT utilizado comporta até 10.000.000 de clientes conectados simultaneamente (HIVEMQ, 2017), fica claro que o principal gargalo é o próprio banco de dados, já que operações de inserção são relativamente custosas e frequentes no perfil desta aplicação. De acordo com *benchmarks* providenciados pelo time do TimescaleDB (TIMESCALE, 2017),

a solução deles comporta uma taxa de inserção de 111.000 linhas por segundo. Como os dispositivos instalados no laboratório costumam enviar cerca de 3 medições por segundo, com um único banco de dados, o sistema PoliLab seria capaz de comportar cerca de 37.000 dispositivos simultaneamente. Considerando que um laboratório teria em média cerca de 15 dispositivos, e usando um fator de segurança de 2x, isso equivale a aproximadamente 1230 laboratórios. Usando o Laboratório Digital da Poli como base para o cálculo do número médio de alunos por ano, tal sistema comportaria aproximadamente 43.500 alunos, com capacidade de ser replicado caso surja necessidade. Essa replicação pode ser realizada por regiões geográficas, ou escalonando o banco de dados dentro de um mesmo *deployment*.

Figura 19: Comparativo de taxa de ingestão de dados de séries temporais entre PostgreSQL e TimescaleDB



Fonte: (TIMESCALE, 2017)

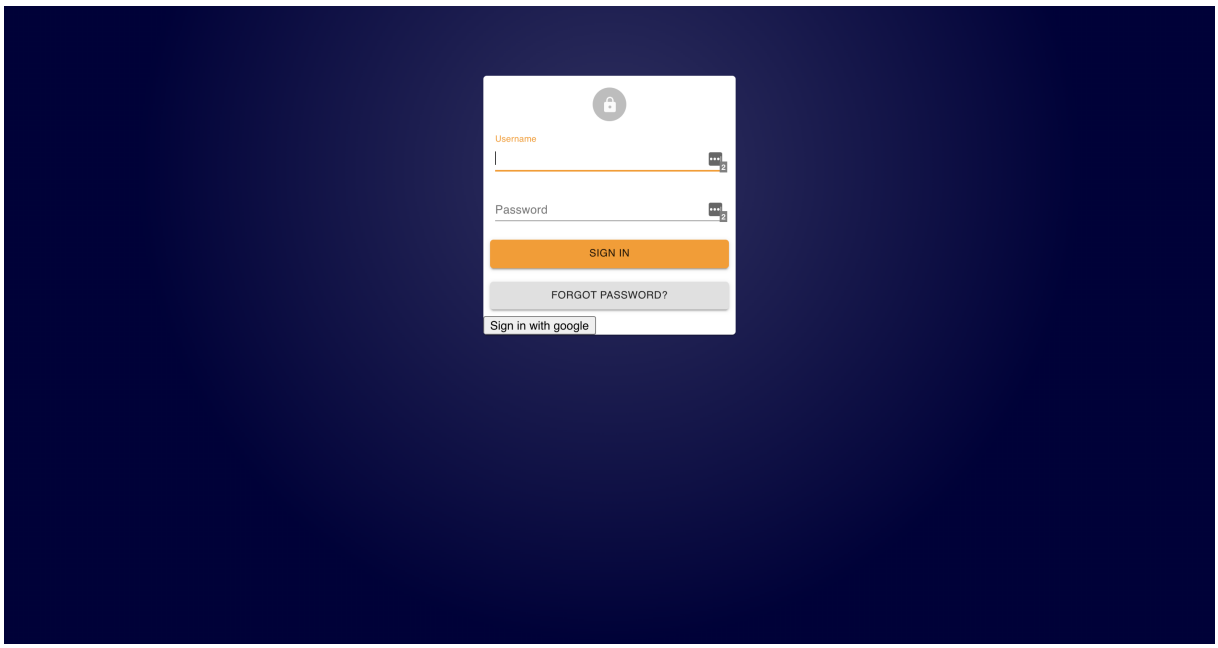
Aplicado ao Laboratório Digital da Poli, o sistema PoliLab permite que funcionários e professores possam controlar e monitorar remotamente o uso e consumo energético dos equipamentos dos laboratórios.

Finalizados a implementação de todos subsistemas, foi necessário criar uma interface

gráfica para que os operadores pudessem interagir de maneira intuitiva com o sistema de gerenciamento.

A Figura 20 mostra a tela de login da aplicação, onde o usuário insere seu email e senha para se comunicar com o subsistema de autenticação. É possível trocar a senha caso o usuário tenha esquecido utilizando o botão de *Forgot Password*. O subsistema de autenticação garante que apenas usuários com emails com domínio @usp.br são capazes de realizar o login, garantindo que o usuário acessando o portal é funcionário ou aluno da USP.

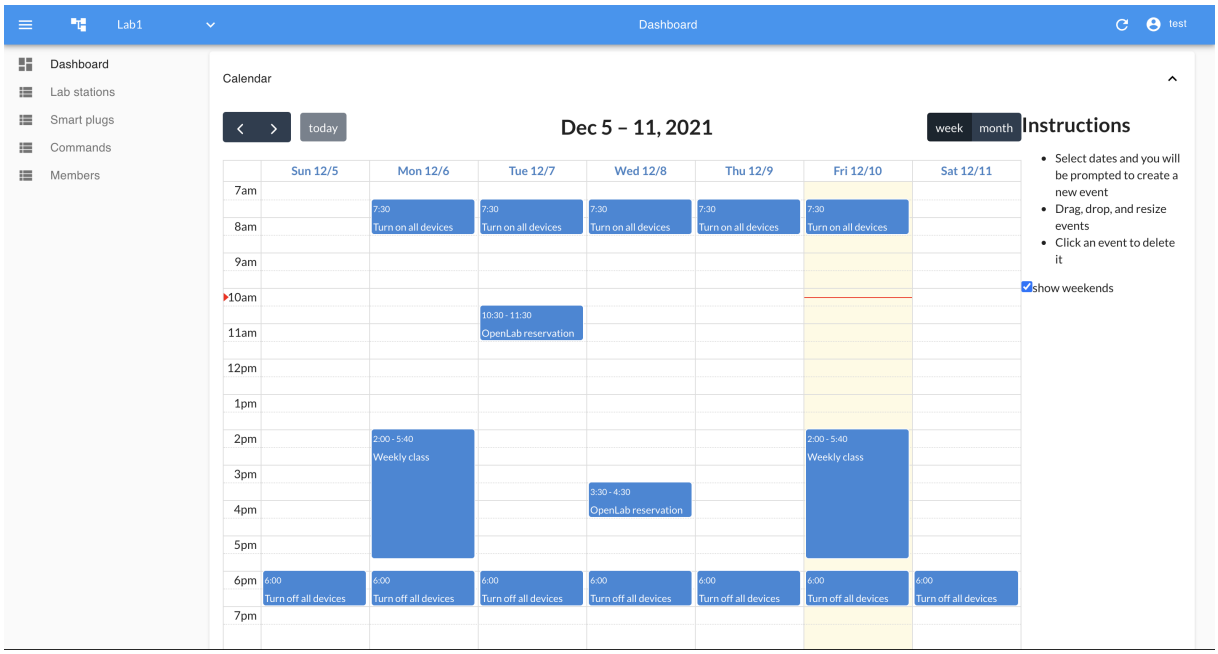
Figura 20: Tela de autenticação da aplicação PoliLab



Fonte: Autoria própria

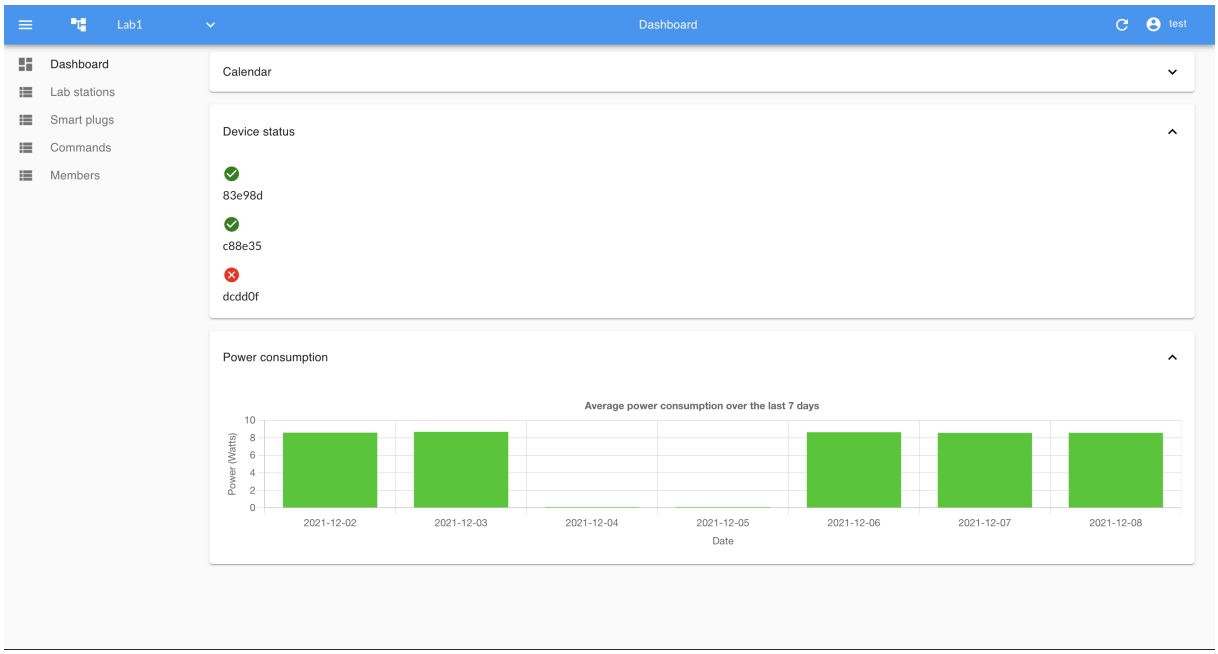
Realizado o login, o usuário se deparará com a dashboard na página inicial. A implementação dessa dashboard foi inspirada no projeto da Dashboard MQTT desenvolvido em (HAYASHI et al., 2021), onde havia *feedback* em tempo real dos eventos ocorridos dentro do laboratório. A dashboard tem 3 componentes principais: um calendário mostrando todos eventos agendados para o laboratório (Figura 21), uma lista de dispositivos em estado operacional que é atualizada em tempo real (Figura 22) e um gráfico mostrando o consumo energético do laboratório na última semana (Figura 22). Esses componentes são integrados com os subsistemas de monitoramento e comandos para providenciar essas informações ao usuário da aplicação.

Figura 21: Calendário de eventos programados para o laboratório "Lab1"da aplicação PoliLab.



Fonte: Autoria própria

Figura 22: Consumo energético dos últimos 7 dias do laboratório "Lab1"da aplicação PoliLab.

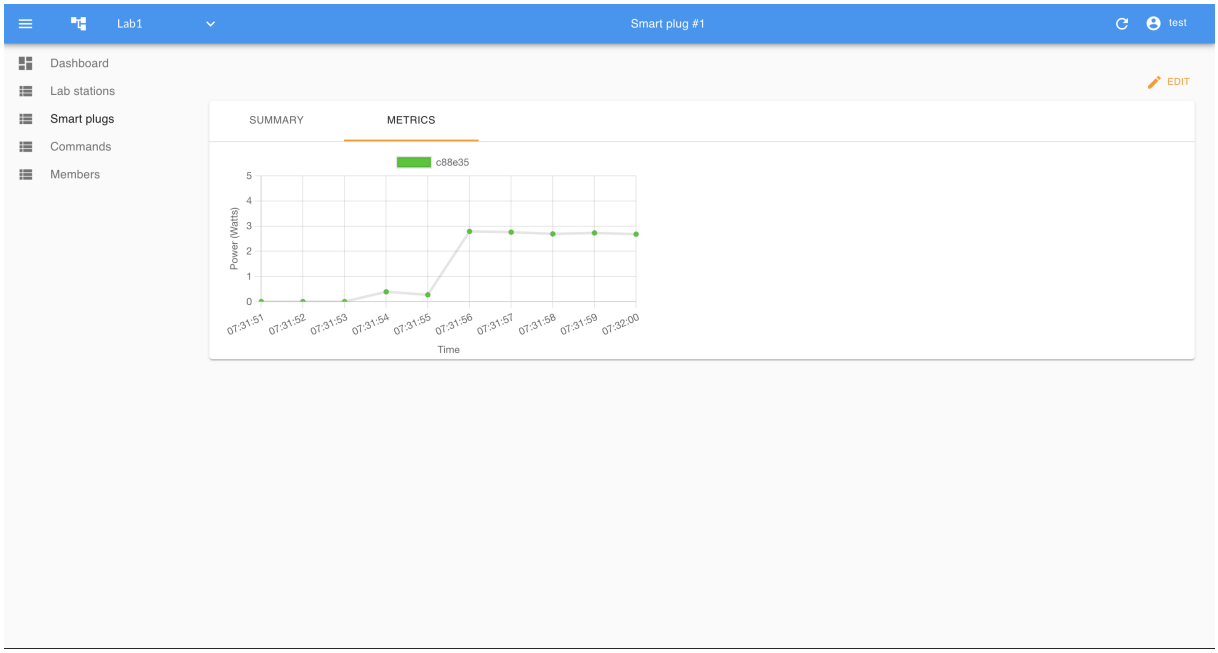


Fonte: Autoria própria

Além da visão geral da dashboard, também é possível extrair métricas de consumo energético de dispositivos específicos em tempo real (Figura 23). Para fazer isso, é neces-

sário acessar a aba de *Smart plugs* no canto superior esquerdo da aplicação e clicar em um dos dispositivos disponíveis na lista.

Figura 23: Métricas de consumo energético do dispositivo "c88e35"em tempo real, monitorado pela aplicação PoliLab



Fonte: Autoria própria

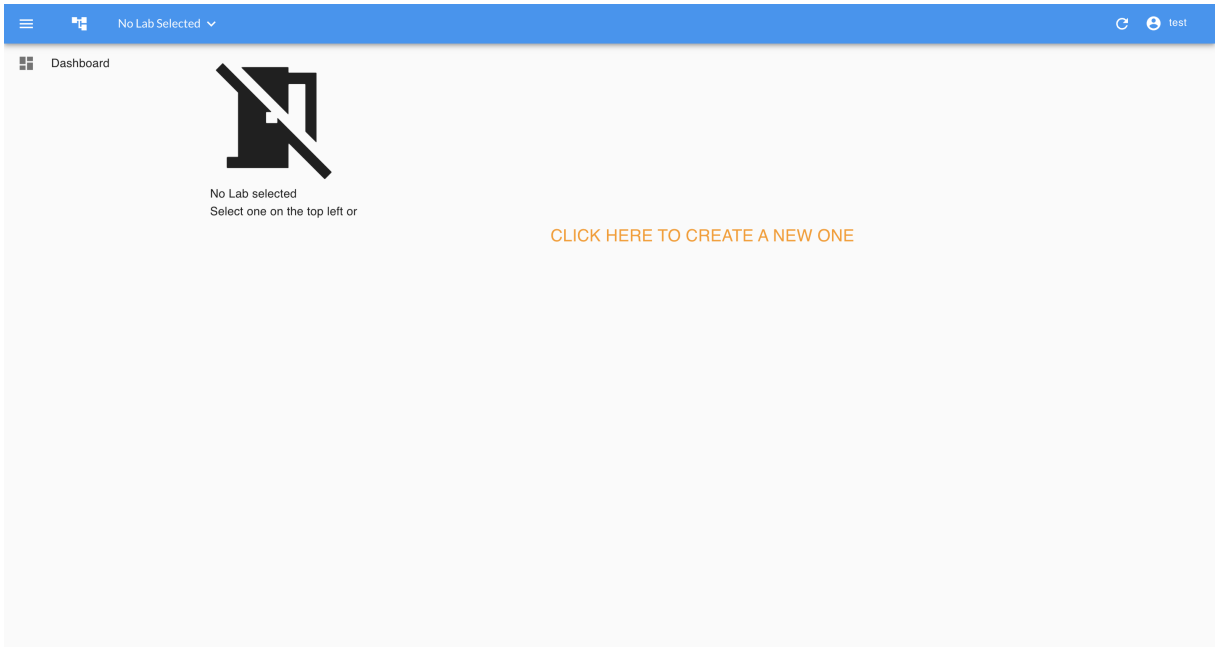
Sendo uma aplicação desenvolvida inteiramente em código aberto, a implementação pode ser encontrada no link <https://github.com/Adarah/TCC>. Para o funcionamento padrão, é necessário ter a ferramenta Docker instalada e um computador com arquitetura x86.

4.1 Configuração da aplicação

Todos os exemplos até agora utilizaram um ambiente pré-configurado para fins de demonstração. A seção abaixo irá detalhar como replicar um ambiente equivalente simulando a jornada de um usuário a partir do seu primeiro acesso.

A primeira barreira que um novo usuário encontrará será a tela de login. Para obter acesso a plataforma, é preciso receber um convite de um usuário (com privilégios elevados para fazer a criação de um convite) ou ser adicionado manualmente por um administrador. Em ambos os casos, o novo membro precisa obter o aval de algum membro já existente, criando uma mini-rede de confiança. Obtido esse acesso, o usuário encontrará a tela inicial mostrada na Figura 24.

Figura 24: Tela de criação de laboratório da aplicação PoliLab



Fonte: Autoria própria

Nesse exemplo, o usuário, que foi adicionado manualmente por um administrador, não é membro de nenhum laboratório e por isso será preciso criar um novo ambiente. Para criar o laboratório, será preciso fornecer um conjunto de informações básicas como o nome do laboratório, a localização dele (usado para consulta dos alunos), e uma breve descrição do papel deste novo laboratório como visto na Figura 25.

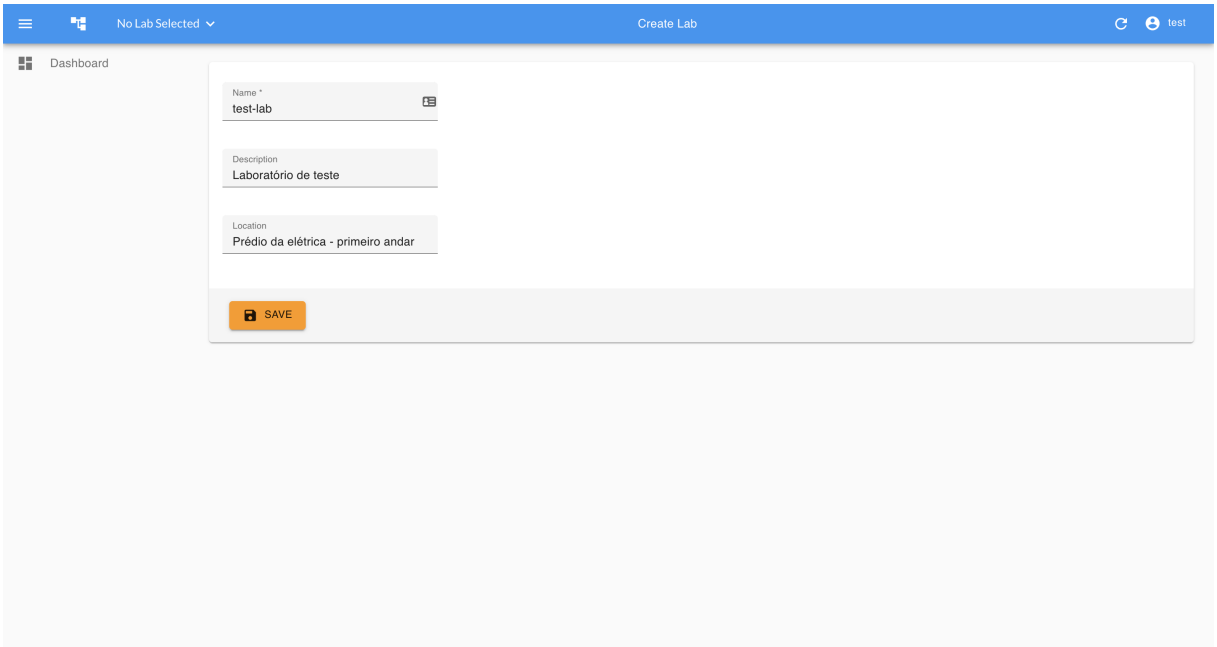


Figura 25: Formulário para criação de um novo laboratório da aplicação PoliLab

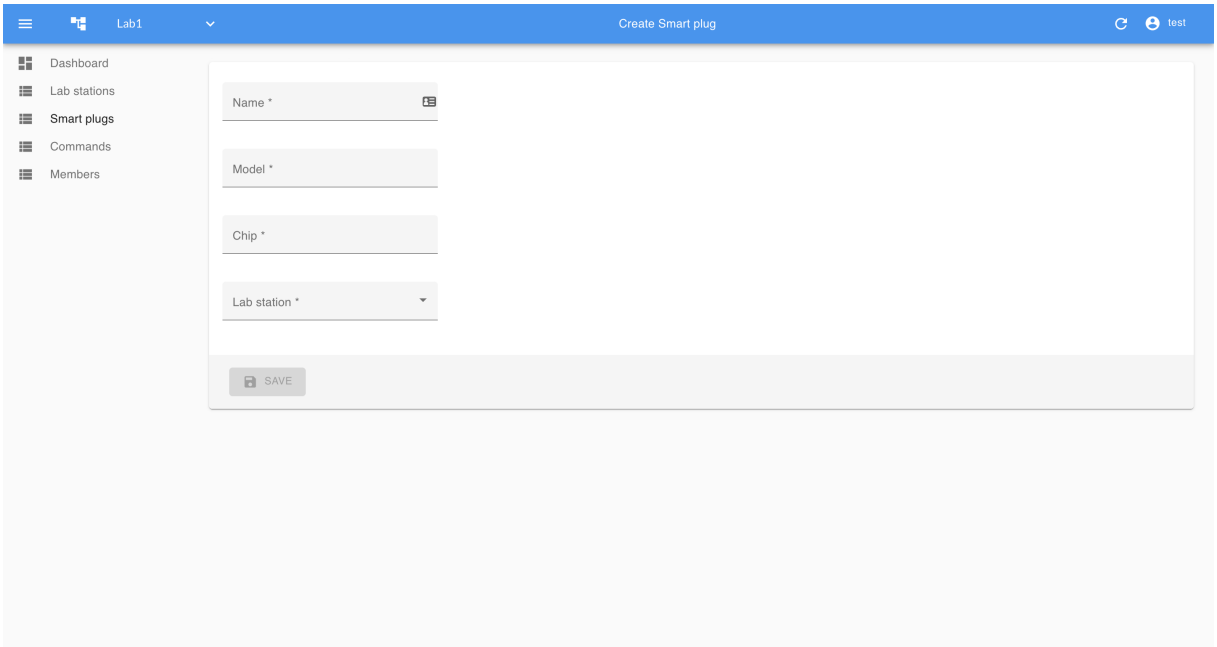
Fonte: Autoria própria

Depois de registrado o laboratório, será preciso fazer o cadastro das bancadas e dispositivos existentes no laboratório. O conceito de "bancada"foi introduzido para permitir fazer um agrupamento lógico entre dispositivos de maneira intuitiva para os funcionários do laboratório, pois o conceito de *tags* arbitrárias acaba sendo um tanto abstrato. A Figura 26 mostra o formulário para o registro de um dispositivo. É preciso fornecer informações como o apelido do dispositivo (usado puramente para facilitar o gerenciamento), o modelo do hardware (no momento apenas só há suporte para um modelo), o número de identificação do hardware, e a bancada que o dispositivo irá pertencer.

Assim que o dispositivo é cadastrado, os dados coletados por ele começam a chegar dentro do sistema em tempo real. Os dados de consumo energético podem ser visualizados acessando a página de um dispositivo específico.

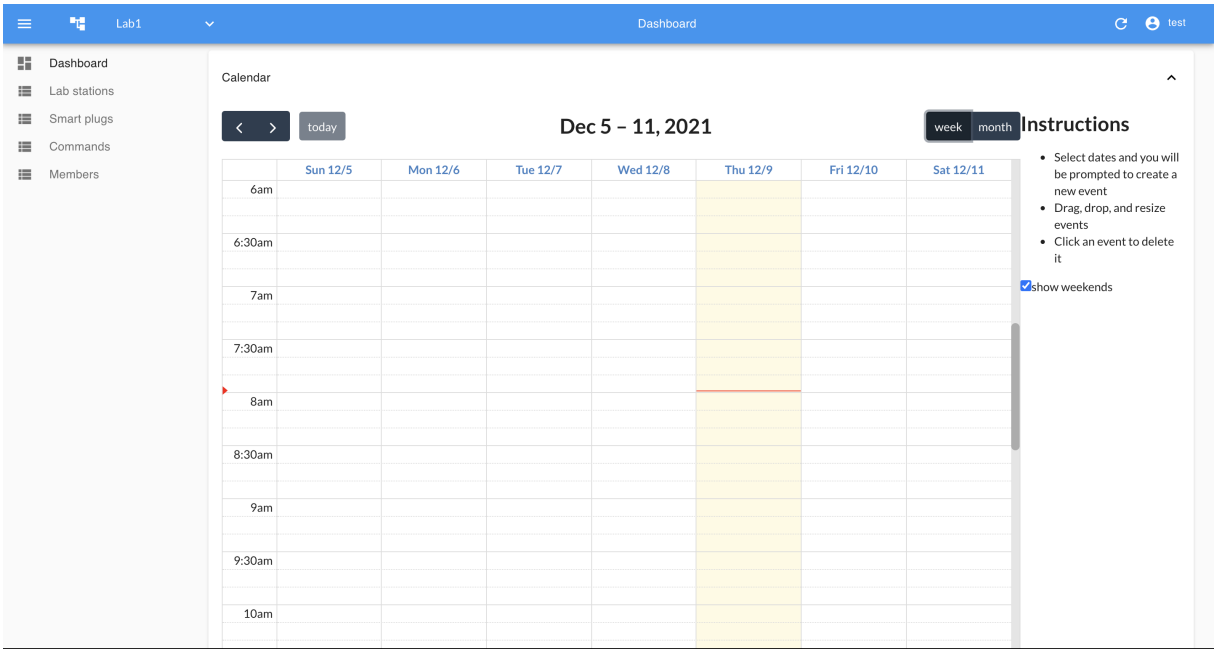
Além da visão detalhada de um dispositivo, depois de haver cadastrado pelo menos um dispositivo com sucesso, a página inicial irá mostrar uma dashboard com métricas do laboratório como um todo, como o consumo energético dos últimos 7 dias, uma lista dos dispositivos em estado operacional, e um calendário dos próximos eventos agendados.

Figura 26: Formulário para registro de um novo dispositivo na aplicação PoliLab



Fonte: Autoria própria

Figura 27: Calendário de eventos programados da aplicação PoliLab. No momento se encontra vazio pois não há comandos registrados.

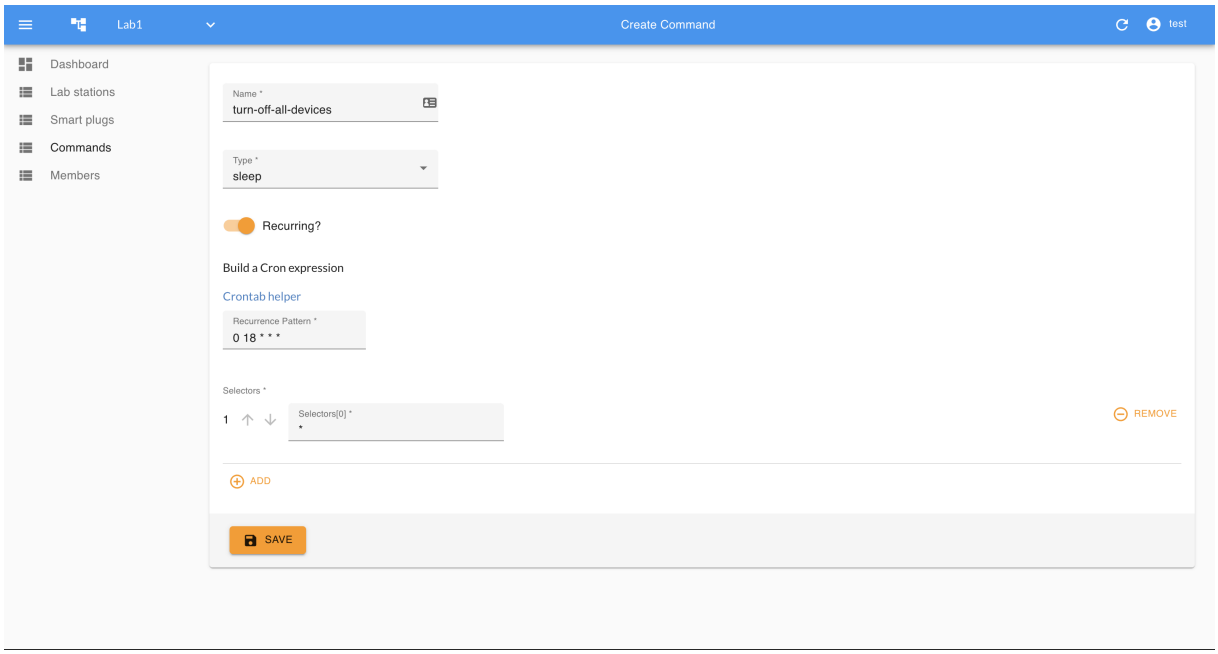


Fonte: Autoria própria

Como visto na dashboard (Figura 27), o laboratório não tem nenhum comando programado por enquanto. O cadastro de um comando pode ser realizado clicando na seção de *Commands* no canto esquerdo da página. Comandos podem ser de uso único ou recor-

rentes.

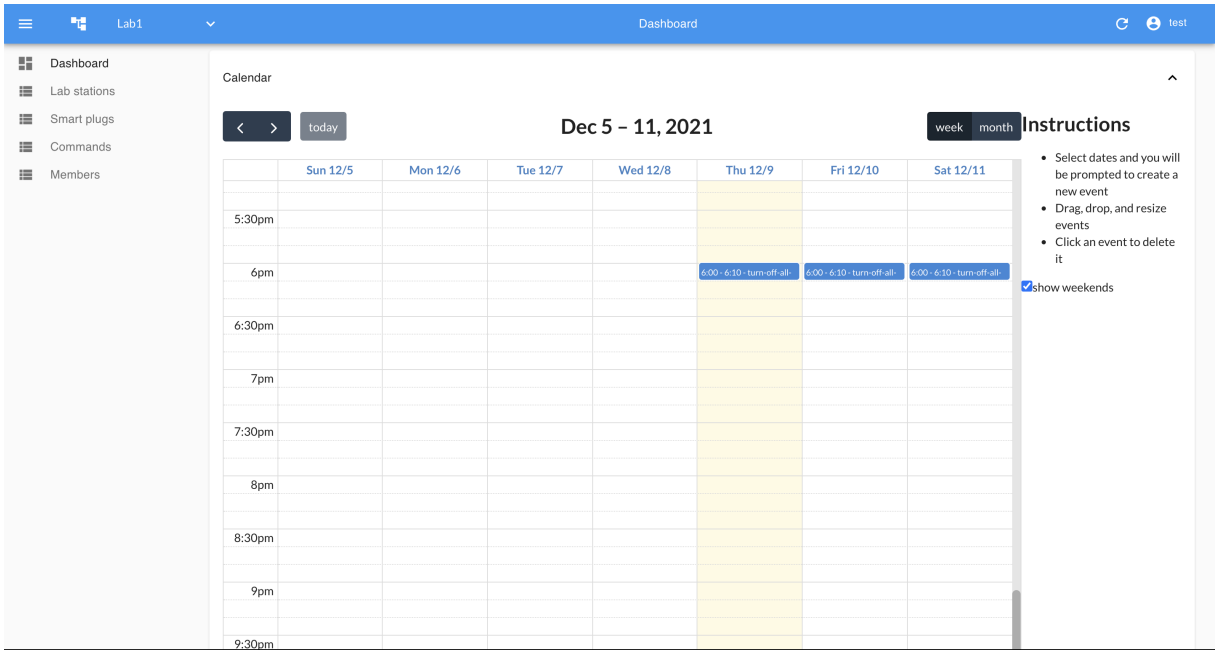
Figura 28: Agendando um comando para desligar todos aparelhos do laboratório todos os dias às 18 horas na aplicação PoliLab.



Fonte: Autoria própria

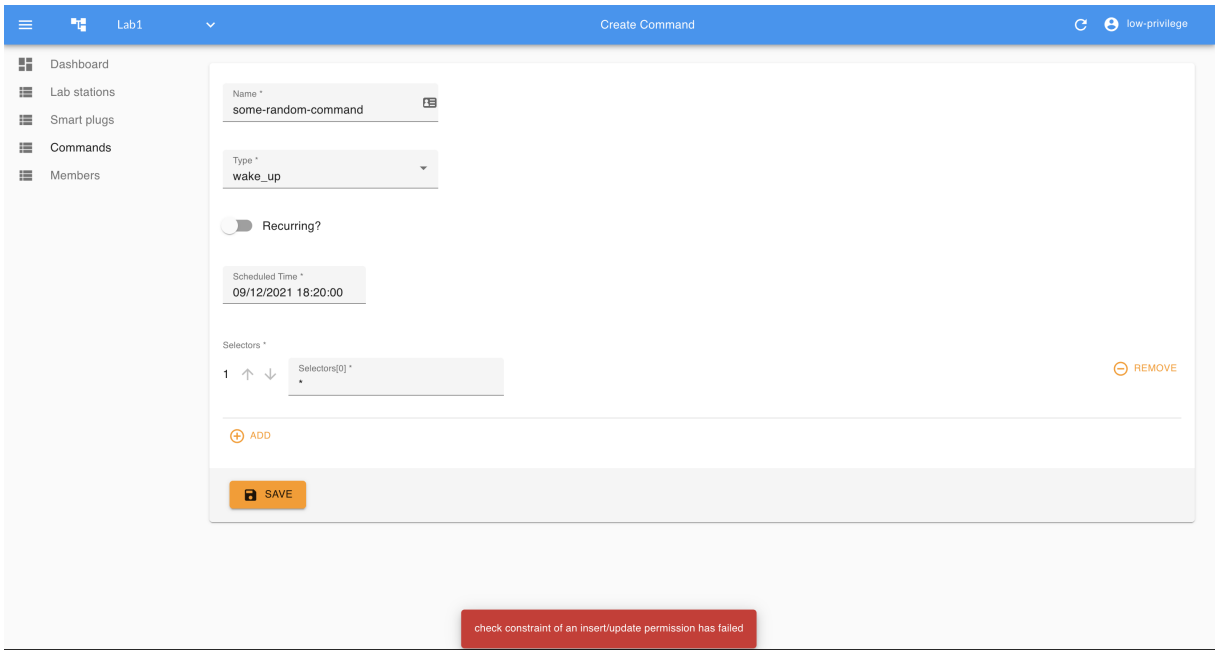
Na Figura 28, está sendo programado um comando que irá desligar todos os equipamentos do laboratório automaticamente todos os dias às 18:00 horas, cujo resultado pode ser visto no calendário da dashboard (Figura 29).

Figura 29: Calendário da aplicação PoliLab atualizado com o novo comando registrado.



Fonte: Autoria própria

Figura 30: Tentativa de criação de comando bloqueada devido a falta de privilégios necessários por parte do usuário



Fonte: Autoria própria

Tentativas de se realizar comandos utilizando um usuário sem privilégios suficientes (Figura 30) são bloqueadas automaticamente e retornam um erro na interface, demonstrando o funcionamento do sistema de autorização.

5 CONCLUSÃO

5.1 Sobre o objetivo do projeto de TCC

O objetivo inicial do projeto foi elaborar e implementar uma arquitetura robusta para facilitar a integração de dispositivos IoT com laboratórios educacionais, partindo de primeiros princípios da identificação de um problema existente e projetando um produto viável comercialmente até a realização da entrega do artefato final. O projeto inicialmente focou nas necessidades do Laboratório Digital da Poli, mas também pode ser utilizado em diversos outros tipo de laboratórios educacionais.

Para demonstrar as funcionalidades implementadas e amparar a usabilidade do produto, foi criada uma interface Web com uma dashboard de monitoramento em tempo real (via Websockets) dos equipamentos instalados no laboratório. A implementação da dashboard foi inspirada no projeto desenvolvido no artigo (HAYASHI et al., 2021) (na qual o autor desta monografia também contribuiu paralelamente ao desenvolvimento deste projeto) que recebeu menção honrosa da banca examinadora. Por meio dessa interface Web, é possível interagir com todos subsistemas necessários para operar os dispositivos IoT, permitindo realizar o agendamento de comandos, registro de novos dispositivos, envio de comandos avulsos, e verificar um calendário com os próximos eventos planejados.

Por conta de restrições de tempo, não foi possível realizar todos os refinamentos planejados. Uma possível melhoria do ponto de vista de engenharia seria a execução de testes de carga para aferir a capacidade de processamento versus a demanda de comandos e coleta de métricas para um devido planejamento de alta disponibilidade. Muito provavelmente, o gargalo de performance estaria na coleta das métricas dos dispositivos já que isso está atrelado a um grande número de inserções por segundo. Por outro lado, o TimescableDB tem foi projeto para cargas de trabalho de séries temporais, e por esse motivo, acredita-se que uma instância é mais do que suficiente para comportar milhares de dispositivos. Os valores numéricos para comprovar tal afirmação podem ser encontrados na seção de Resultados.

Do ponto de vista de hardware, teria sido interessante explorar protocolos de comunicação alternativos. O design do projeto atual assume que existe uma conexão relativamente estável dentro do Laboratório Digital da Poli para que os dispositivos se comuniquem com o *broker* MQTT, mas essa suposição não é universalmente verdade: de acordo com o Censo Escolar de 2018 (INEP CENSO, 2018), 11.7% das escolas da rede estadual com acesso à internet tem conexões abaixo de 128kpbs, o que pode prejudicar principalmente a leitura em tempo real das métricas coletadas. Em tais situações, pode ser realizado um agrupamento das medições em *batches* para se enviá-las uma única vez a cada N horas, a fim de preservar a banda limitada do local.

Em termos de requisitos funcionais e não-funcionais, com exceção do RF04, todos foram atendidos, com um asterisco no RF03. Apesar dele não mencionar explicitamente a necessidade da RFC5545, durante o desenvolvimento do projeto achou-se interessante integrá-la no sistema para permitir o agendamento de comandos e a exportação dos calendários de eventos, mas não foi possível realizar isso a tempo. Isso ocorreu devido a falta de recursos para implementar tal funcionalidade já que não existiam soluções prontas com código aberto no mercado; criar um *scheduler* robusto compatível com a RFC5545 é um projeto que por si só demoraria meses para ser implementado, e portanto se utilizou o formato cron como substituto.

Apesar dos refinamentos anteriores que não foram implementados, pode-se dizer que o objetivo original foi atingido com sucesso: é possível controlar os equipamentos do Laboratório Digital da Poli remotamente, agendar comandos, e monitorar o uso e consumo energético de cada dispositivo individualmente, tudo isso com garantias de autenticação e autorização.

Como toda a implementação foi realizada utilizando código aberto, espera-se que implementações alternativas surjam para suprir as necessidades específicas de cada instituição educacional, mesmo aquelas fora da esfera da USP.

5.2 Em relação ao curso de engenharia

O desenvolvimento deste trabalho exercitou vários aprendizados obtidos durante o curso de engenharia, desde técnicas para organização de projetos como o levantamento de requisitos e a priorização de tarefas, como também práticas de engenharia de software como a realização de um projeto de arquiteturas robustas, extensibilidade e facilidade de manutenção de código, e uso de tecnologias modernas.

Dentro das mais de 250 horas de aula acumuladas durante o curso de engenharia da computação, as disciplinas da Poli cujo aprendizado mais se destacou durante o andar do projeto foram Sistemas Tolerantes a Falhas (PCS3578), Sistemas Digitais (PCS3115, PCS3225), Laboratório Digital (PCS335), Engenharia de Software e Bancos de Dados (PCS3413), Laboratório de Engenharia de software (PCS3343, PCS3553), Gerência e Qualidade de Software (PCS 3563), Redes de computadores (PTC3360, PCS3414, PCS3424), demonstrando o papel multidisciplinar que um engenheiro deve exercer no desenvolvimento de um projeto IoT. Para engenheiros entrando no mercado, a área de IoT representa uma oportunidade riquíssima para se explorar os mais diversos campos de estudo, pois combina conhecimentos de hardware, software, sistemas distribuídos, segurança, e até aprendizado de máquina.

5.3 Possíveis evoluções do produto

Existem várias possibilidades para dar continuidade ao projeto. Exemplos incluem (a) integrar o módulo de autenticação diretamente com sistema de autenticação da USP via SAML (*Security Assertion Markup Language*), (b) permitir exportar calendários dos laboratórios para formato RFC 5545 iCal, o que permitiria integrar com ferramentas como o Google Calendar e o iCloud Calendar, (c) integrar o módulo de comandos diretamente com o sistema e-disciplinas para permitir criação de reservas de horários diretamente no website, (d) integrar o módulo de monitoramento com o e-disciplinas para permitir visualizar estado dos laboratórios no próprio website, (e) utilizar dados de consumo de energia e engajamento dos alunos gerados pela aplicação para fazer análises e aplicar métodos estatísticos como Machine Learning, por exemplo, poderia se decidir analiticamente quais os melhores horários para se enviar comandos, (f) utilização de técnicas de Business Inteligence para obter um entendimento mais profundo do comportamento dos alunos e do consumo energético dos equipamentos.

REFERÊNCIAS

ACHUTHAN, K. et al. The value @ amrita virtual labs project: Using web technology to provide virtual laboratory access to students. In: *2011 IEEE Global Humanitarian Technology Conference*. [S.l.: s.n.], 2011. p. 117–121.

AKTAN, B. et al. Distance learning applied to control engineering laboratories. *IEEE Transactions on Education*, v. 39, n. 3, p. 320–326, 1996.

ALMEIDA, F. et al. Laboratório digital à distância: Percepções de docentes e discentes. In: SBC. *Anais do Simpósio Brasileiro de Educação em Computação*. [S.l.], 2021. p. 316–325.

ARAKAKI, R.; HAYASHI, V. T.; RUGGIERO, W. V. Available and fault tolerant iot system: Applying quality engineering method. In: *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*. [S.l.: s.n.], 2020. p. 1–6.

ASHTON, K. et al. That 'internet of things' thing. *RFID journal*, v. 22, n. 7, p. 97–114, 2009.

BALAMURALITHARA, B.; WOODS, P. C. Virtual laboratories in engineering education: The simulation lab and remote lab. *Computer Applications in Engineering Education*, Wiley Online Library, v. 17, n. 1, p. 108–118, 2009.

BENCOMO, S. D. Control learning: Present and future. *Annual Reviews in control*, Elsevier, v. 28, n. 1, p. 115–136, 2004.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. *Diário Oficial [da] República Federativa do Brasil*, Brasília, DF, 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>.

BREWER, E. A. Towards robust distributed systems. In: PORTLAND, OR. *PODC*. [S.l.], 2000. v. 7, n. 10.1145, p. 343477–343502.

CHEN, S. et al. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things Journal*, v. 1, n. 4, p. 349–359, 2014.

CRUZ, G. B. d.; CAMPOS, J. B. d. C. Laboratórios. 2016.

DACHYAR, M.; ZAGLOEL, T. Y. M.; SARAGIH, L. R. Knowledge growth and development: internet of things (iot) research, 2006–2018. *Heliyon*, v. 5, n. 8, p. e02264, 2019. ISSN 2405-8440. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2405844019359249>.

EUROPEAN COMMISSION. *2018 reform of EU data protection rules*. 2018. Disponível em: <https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679>.

FARHAN, L. et al. A survey on the challenges and opportunities of the internet of things (iot). In: *2017 Eleventh International Conference on Sensing Technology (ICST)*. [S.l.: s.n.], 2017. p. 1–5.

FEISEL, L.; PETERSON, G. A colloquy on learning objectives for engineering education laboratories. In: *2002 Annual Conference*. [S.l.: s.n.], 2002. p. 7–20.

FEISEL, L. D.; ROSA, A. J. The role of the laboratory in undergraduate engineering education. *Journal of engineering Education*, Wiley Online Library, v. 94, n. 1, p. 121–130, 2005.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine, 2000.

FIELDPRO. *Fieldpro*. 2021. Disponível em: <https://fieldpro.com.br>.

GEORGIOU, J.; DIMITROPOULOS, K.; MANITSARIS, A. A virtual reality laboratory for distance education in chemistry. *International Journal of Social Sciences*, v. 2, n. 1, p. 34–41, 2007.

GRAVIER, C. et al. State of the art about remote laboratories paradigms–foundations of ongoing mutations. *International Journal of Online Engineering*, v. 4, n. 1, 2008.

HAERDER, T.; REUTER, A. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 15, n. 4, p. 287–317, dez. 1983. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/289.291>.

HALLER, S.; KARNOUSKOS, S.; SCHROTH, C. The internet of things in an enterprise context. In: SPRINGER. *Future internet symposium*. [S.l.], 2008. p. 14–28.

HARDT, D. et al. *The OAuth 2.0 authorization framework*. [S.l.]: RFC 6749, October, 2012.

HAYASHI, V. et al. Labead: Laboratório remoto para o ensino de engenharia. In: *Anais dos Workshops do IX Congresso Brasileiro de Informática na Educação*. Porto Alegre, RS, Brasil: SBC, 2020. p. 187–194. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/wcbie/article/view/13044>.

HAYASHI, V.; ALMEIDA, F.; KOMO, A. Labbitcoin: Fpga iot testbed for bitcoin experiment with energy consumption. In: *Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2021. p. 90–97. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/17344>.

HAYASHI, V.; HAYASHI, F. Labead: Laboratório eletrônico de ensino à distância durante o distanciamento social. In: *Anais do V Congresso sobre Tecnologias na Educação*. Porto Alegre, RS, Brasil: SBC, 2020. p. 21–30. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/ctrle/article/view/11379>.

HAYASHI, V.; RUGGIERO, W. Non-invasive challenge response authentication for voice transactions with smart home behavior. *Sensors*, v. 20, n. 22, 2020. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/20/22/6563>.

HAYASHI, V. T. et al. Dashboard iot remote lab with mqtt protocol. In: *Anais Estendidos do XXVII Simpósio Brasileiro de Sistemas Multimídia e Web*. Porto Alegre, RS, Brasil: SBC, 2021. p. 67–70. ISSN 2596-1683. Disponível em: <https://sol.sbc.org.br/index.php/webmedia_estendido/article/view/17614>.

HAYASHI, V. T.; HAYASHI, F. H.; ARAKAKI, R. Labead: Laboratório interativo para o ensino de eletrônica durante a covid-19. *Brazilian Journal of Development*, v. 6, n. 9, p. 72600–72620, 2020.

HERADIO, R. et al. Virtual and remote labs in education: A bibliometric analysis. *Computers & Education*, Elsevier, v. 98, p. 14–38, 2016.

HIVEMQ. *10,000,000 MQTT Clients*. 2017. Disponível em: <https://www.hivemq.com/benchmark-10-million/>.

INEP CENSO. notas estatísticas. *Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira*, p. 15–44, 2018.

MIRANDA, P.; SIEKKINEN, M.; WARIS, H. Tls and energy consumption on a mobile device: A measurement study. In: *2011 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.: s.n.], 2011. p. 983–989.

OSTERWALDER, A. et al. *Value proposition design: How to create products and services customers want*. [S.l.]: John Wiley & Sons, 2014. v. 2.

PATEL, K. K.; PATEL, S. M. et al. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, v. 6, n. 5, 2016.

REYNA, A. et al. On blockchain and its integration with iot. challenges and opportunities. *Future generation computer systems*, Elsevier, v. 88, p. 173–190, 2018.

SHAFIQ, M. et al. Corrauc: a malicious bot-iot traffic detection method in iot network using machine-learning techniques. *IEEE Internet of Things Journal*, IEEE, v. 8, n. 5, p. 3242–3254, 2020.

SHI, W. et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, IEEE, v. 3, n. 5, p. 637–646, 2016.

STACK OVERFLOW. *2021 developer survey*. 2021. Disponível em: <https://insights.stackoverflow.com/survey/2021>.

SURESH, P. et al. A state of the art review on the internet of things (iot) history, technology and fields of deployment. In: *2014 International Conference on Science Engineering and Management Research (ICSEMR)*. [S.l.: s.n.], 2014. p. 1–8.

TAELMAN, R.; SANDE, M. V.; VERBORGH, R. Graphql-ld: linked data querying with graphql. In: *ISWC2018, the 17th International Semantic Web Conference*. [S.l.: s.n.], 2018. p. 1–4.

TIMESCALE. *TimescaleDB vs. PostgreSQL for time-series*. 2017. Disponível em: <https://blog.timescale.com/blog/timescaledb-vs-6a696248104e/>.

TOKEN, J. W. Internet engineering task force std. *RFC7519*, 2015.

USP. *Informações da Disciplina de Laboratórios Digitais*. 2021. Disponível em: <https://uspdigital.usp.br/jupiterweb/obterDisciplina?sgldis=PCS3335>.

VERMA, A.; RANGA, V. Machine learning based intrusion detection systems for iot applications. *Wireless Personal Communications*, Springer, v. 111, n. 4, p. 2287–2310, 2020.

APÊNDICE A – ARQUIVOS DE CONFIGURAÇÃO

Abaixo se encontra as configurações das tecnologias utilizadas no projeto. Valores fazendo referência a *localhost* devem ser alterados para o domínio usado no *deployment* da aplicação. Certos campos foram omitidos para evitar vazamento de usuários e senhas.

```
{
    "hasura": {
        "shared_secret": "<OMITTED>",
        "backend_url": "http://localhost:3001"
    }
}
```

Listing 2: Firebase runtime config

```
MQTT_BROKER_URL=<OMITTED>
MQTT_BROKER_USERNAME=<OMITTED>
MQTT_BROKER_PASSWORD=<OMITTED>
PG_DATABASE=poli_lab
PG_USER=<OMITTED>
PG_PASSWORD=<OMITTED>
```

Listing 3: Configuração do sistema de monitoramento

```
NODE_ENV=development
MQTT_BROKER_ADMIN_USERNAME=<OMITTED>
MQTT_BROKER_ADMIN_PASSWORD=<OMITTED>
FIREBASE_AUTH_EMULATOR_HOST=localhost:9099
SELF_HOST=localhost
```

Listing 4: Configuração do sistema de comandos

```yaml
version: '3.6'
services:
  timescale:
    image: timescale/timescaledb:latest-pg12
    restart: always
    environment:
      POSTGRES_PASSWORD: <OMITTED>
    ports:
      - "5432:5432"
    command:
    - postgres
    - -c
    - "shared_preload_libraries=timescaledb"
    volumes:
    - db_data:/var/lib/postgresql/data
    - ./postgres-initdb:/docker-entrypoint-initdb.d:Z
  graphql-engine:
    image: hasura/graphql-engine:v2.0.6.cli-migrations-v3
    ports:
    - "8080:8080"
    depends_on:
    - "timescale"
    restart: always
    environment:
      HASURA_GRAPHQL_DATABASE_URL:
      ↪ postgres://<OMITTED>:<OMITTED>@timescale:5432/poli_lab
      # enable the console served by server
      HASURA_GRAPHQL_ENABLE_CONSOLE: "false" # set to "false" to
      ↪ disable console
      # enable debugging mode. It is recommended to disable this in
      ↪ production
      HASURA_GRAPHQL_ENABLED_LOG_TYPES: startup, http-log,
      ↪ webhook-log, websocket-log, query-log
      # uncomment next line to set an admin secret
      HASURA_GRAPHQL_ADMIN_SECRET: <OMITTED>
      HASURA_GRAPHQL_AUTH_HOOK:
      ↪ http://host.docker.internal:3001/authWebhookHandler
    volumes:
    - ./migrations:/hasura-migrations:Z
    - ./metadata:/hasura-metadata:Z
volumes:
  db_data:
```

Listing 5: Hasura docker-compose config